Implementation and validation of an Oxygen Exchange model coupled with multiple physics

life^x project

Project Report for

Advanced Programming for Scientific Computing

A.Y. 2020 - 2021

Teresa Babini

Manfred Nesti

Supervisors:

Dr. P. C. Africa, Dr. I. Fumagalli, Dr. F. Regazzoni

POLITECNICO
MILANO 1863

Contents

Chapter 1

Introduction

The **iHEART** project [9], coordinated by prof. A. Quarteroni, represents one of the first attempts in the world to create a complete mathematical model of the human heart, which includes all the physiological processes that together form the complexity that we call life. The ultimate goal is to build a virtual model of the heart capable of helping clinicians study the genesis and treatment of cardiovascular diseases, improving prevention, treatment and cardiac surgery.
The proposed models are implemented in **life**^{x} library [7], which is a high-performance Finite Element library written in C++ and based on the deal.II library as finite element core.
The aim of our work, is to implement a model for the **Oxygen Exchange (OE)** to the cardiac muscle tissues (myocardium) and to couple it with other models in order to see how some patient-specific conditions influence the oxygen concentration and pressure in the myocardium. We made an effort to implement our contributions in a way as consistent as possible with the existing code. The developed features are currently under review to be merged into **life**^{x} master branch (see `https://gitlab.com/lifex/lifex/-/merge_requests/291`).
External respiration is the diffusion of O2 from air in the alveoli of the lungs to blood in pulmonary capillaries and converts deoxygenated blood coming from the right side of the heart into oxygenated blood that returns to the left side of the heart. The myocardium requires continuous oxygen supply to properly contract and pump the blood into the whole arterial system. Oxygenated blood reaches the heart through the coronaries and perfuses the whole myocardium, allowing the exchange of oxygen at the level of the microvasculature.
A reduced oxygen delivery to the myocardium can occur because of various reasons and it is important to analyse the dynamics of the concentration of oxygen and its partial pressure inside the myocardium muscle. The model we implemented has the intent of giving as output, together with the spatial distribution of the oxygen uptake and partial oxygen pressure in the muscle, also the total amount of oxygen delivered to the muscle, the total amount of oxygen

consumed by it and an upper bound to the averaged oxygen flux. Finally it analyzes how those features change depending on specific patient parameters or perfusion inputs.
This work is structured as follows. We start giving a briefly description of the physical derivation of the OE model and of its mathematical formulation. Afterwards, we describe the **life**^{x} library and how our code is organized and implemented. Finally, we present the results we obtained both with some possible future developments of the model.

Chapter 2

Mathematical Framework

The model describing the change of oxygen concentration and pressure in the myocardium was developed in [1]. In this section we will give an insight on the model we implemented and the physical quantities recovered, then we focuses on the discretization of the equations to solve through FEM (Finite Element Method), giving an overview on the 0D and 3D model scheme and mathematical tools used to solve them.

2.1 Oxygen Exchange (OE) model

In this section we firstly show the model as it is, describing the dynamics of oxygen saturation in the blood SO_2, partial pressure of oxygen in the blood PO_2 and partial pressure of oxygen in the myocardium PO_2^m. Then, we give a brief discussion on the parameters involved in the equations, in particular the quantities whose dynamics are described by the myocardial perfusion. Finally, we recover a reduced model, computationally more affordable, whose unknowns are the total oxygen concentration $[O_2*]$ and again the partial pressure of oxygen in the myocardium PO_2^m.

2.1.1 Complete model

We consider the myocardium to be divided into 3 compartments. Let us denote

- ψ_i and ψ_m [dimensionless] the volume fraction occupied by the compartment $i = 1, 2, 3$ and occupied by the muscle respectively, so we have $\psi_1 + \psi_2 + \psi_3 + \psi_m = 1$;
- ρ_{blood} [kg m^{-3}] the effective density (mass per volume of tissue) of the blood contained in the compartment i.
- $\Phi_{i,j}$ [kg m^{-3} s^{-1}] the mass flux between compartment $i, j = 1, 2, 3$.

The three compartments occupy an identical volume fraction in each region of space, which does not change as time goes by: this means that the vascular walls are rigid.

Starting from oxygen-hemoglobin association/dissociation chemical reaction

$$\mathrm{Hb} + n\,\mathrm{O}_2 \underset{k_-}{\overset{k_+}{\rightleftharpoons}} \mathrm{Hb} - \mathrm{O}_2 \tag{2.1}$$

we can derive an equation describing the mass conservation of the quantity of oxygen. Experimental curves find that hemoglobin has $n \simeq 3$ binding sites for oxygen.
The amount of oxygen stored within the compartment i inside the volume V is given by $\int_V c^i_{\mathrm{O}_2}$, where we denote by c_{O_2} [mol m^{-3}] the overall concentration of O_2 in the tissue (moles per unit volume of tissue). The rate of variation of this quantity is the result of

- chemical reactions;
- advection;
- diffusion;
- flux from the neighboring compartments;
- flux outgoing from the considered compartment;
- oxygen exchanged with the muscle (assume that oxygen is exchanged only by the third compartment).

Assuming that blood always flows downstream, i.e. $\phi_{i-1,i} \geqslant 0$ for any i, we get O_2

$$\begin{aligned} \frac{\partial}{\partial t}[\mathrm{O}_2]^i &= n\left(k_-[\text{Hb–O2}]^i - k_+[\mathrm{Hb}]^i\left([\mathrm{O}_2]^i\right)^n\right) - \nabla\cdot\left([\mathrm{O}_2]^i\,\mathbf{u}_i\right) \\ &+ \phi_{i-1,i}[\mathrm{O}_2]^{i-1} - \phi_{i,1+i}[\mathrm{O}_2]^{i+1} - \psi_i^{-1}\frac{A_v^i}{V_t}P\left([\mathrm{O}_2]^i - [\mathrm{O}_2]^m\right)\delta_{i3} \end{aligned} \tag{2.2}$$

where $[A]$ (moles per unit volume of fluid) denotes the concentration of quantity A in the blood. By proceeding in the same manner for hemoglobin and oxyhemoglobin (with the only difference that we assume that the vascular membrane is not permeable to them), we get the analogous equations

$$\begin{aligned} \frac{\partial}{\partial t}[\mathrm{Hb}]^i &= \left(k_-[\text{Hb–O2}]^i - k_+[\mathrm{Hb}]^i\left([\mathrm{O}_2]^i\right)^n\right) - \nabla\cdot\left([\mathrm{Hb}]^i\,\mathbf{u}_i\right) \\ &+ \phi_{i-1,i}[\mathrm{Hb}]^{i-1} - \phi_{i,i+1}[\mathrm{Hb}]^i \\ \frac{\partial}{\partial t}[\text{Hb–O2}]^i &= -\left(k_-[\text{Hb–O2}]^i - k_+[\mathrm{Hb}]^i\left([\mathrm{O}_2]^i\right)^n\right) - \nabla\cdot\left([\text{Hb–O2}]^i\,\mathbf{u}_i\right) \\ &+ \phi_{i-1,i}[\text{Hb–O2}]^{i-1} - \phi_{i,i+1}[\text{Hb–O2}]^i \end{aligned} \tag{2.3}$$

Since oxygen saturation is defined as $\mathrm{SO}_2^i = [\text{Hb–O2}]^i/[\mathrm{Hb}^*]$, we have that

- $[\text{Hb–O2}]^i = {\text{SO}_2}^i\,[\text{Hb}^*]$;
- $[\text{Hb}]^i = \left(1 - {\text{SO}_2}^i\right)[\text{Hb}^*]$

Moreover, by the Henry law, the partial pressure of Oxygen is given by

$$ {\text{PO}_2}^i = \alpha\,[\text{O}_2]^i \tag{2.4} $$

Defining the total concentration of hemoglobin as $[\text{Hb}^*]^i := [\text{Hb}]^i + [\text{Hb–O2}]^i$ and summing the two lines of (2.3), we get

$$ \frac{\partial}{\partial t}[\text{Hb}^*]^i + \nabla\cdot\left([\text{Hb}^*]\,\mathbf{u}_i\right) = \phi_{i-1,i}\,[\text{Hb}^*]^{i-1} - \phi_{i,i+1}\,[\text{Hb}^*]^i \tag{2.5} $$

This, together with the continuity equation for blood velocity $\mathbf{u}$

$$ \nabla\cdot\mathbf{u}_i = \Phi_{i-1,i} - \Phi_{i,i+1} $$

entails that $[\text{Hb}^*]^i$ is constant in space and time and equal for each compartment. This is consistent with the hypothesis of the model, as hemoglobin is never exchanged with the surrounding environment. Therefore, we henceforth drop the apex i and we consider $[\text{Hb}^*]$ as constant.

In conclusion, using ${\text{SO}_2}^i$ and ${\text{PO}_2}^i$ as primary variables, we obtain, for $i = 1, 2, 3$ the equations

$$ \begin{aligned} \frac{\partial}{\partial t}{\text{PO}_2}^i + \nabla\cdot\left({\text{PO}_2}^i\,\mathbf{u}_i\right) &= n\alpha\,[\text{Hb}\,*]\left(k_-\,{\text{SO}_2}^i - k_+\alpha^{-n}\left(1 - {\text{SO}_2}^i\right)\left(({\text{PO}_2}^i)^n\right)\right) \\ &\quad + \psi_i^{-1}\phi_{i-1,i}\,{\text{PO}_2}^{i-1} - \phi_{i,i+1}\,{\text{PO}_2}^i \\ &\quad - \psi_i^{-1}\frac{A_v^i}{V_t}P\left({\text{PO}_2}^i - {\text{PO}_2}^m\right)\delta_{i3} \\ \frac{\partial}{\partial t}{\text{SO}_2}^i + \nabla\cdot\left({\text{SO}_2}^i\,\psi_i^{-1}\mathbf{u}_i\right) &= -\left(k_-\,{\text{SO}_2}^i - k_+\alpha^{-n}\left(1 - {\text{SO}_2}^i\right)\left({\text{PO}_2}^i\right)^n\right) \\ &\quad + \phi_{i-1,i}\,{\text{SO}_2}^{i-1} - \phi_{i,i+1}\,{\text{SO}_2}^i \end{aligned} \tag{2.6} $$

The total amount of oxygen delivered to the muscle Λ_{O_2} [mol m^{-3} s^{-1}] can be computed as

$$ \Lambda_{\text{O}_2} = \frac{1}{T|\Omega|}\int_0^T\int_\Omega \alpha^{-1}\frac{A_v^i}{V_t}P\left({\text{PO}_2}^i - {\text{PO}_2}^m\right) \tag{2.7} $$

2.1.2 Oxygen dynamics inside the muscle

So far we have considered the oxygen concentration in the muscle($[\text{O}_2]^m$ and the corresponding partial pressure ${\text{PO}_2}^m = \alpha\,[\text{O}_2]^m$) as a datum. In this section we derive an equation to model its dynamics. We have

$$\frac{d}{dt}\int_V c^m_{O_2} = \int_V \frac{A_v^3}{V_t} P \left([\mathrm{O_2}]^3 - [\mathrm{O_2}]^m \right) - \int_v \psi_m \xi \tag{2.8}$$

where ξ [mol m^{-3}s^{-1}] is a consumption per unit volume of muscle representing the oxygen consumption rate of the metabolic activity. For the latter term we could adopt several choices, with increasing biophysical detail.

- Assuming a constant consumption rate.
- Assuming a Michaelis-Menten dynamics, namely

  $$\xi = \xi_0 \left(1 + \frac{\mathrm{PO_2}^{m,50}}{\mathrm{PO_2}^m} \right)^{-1}$$

  where ξ_0 is the maximum rate and $\mathrm{PO_2}^{m,50}$ is the half maximal effective partial pressure.
- Coupling this model with a model for the cellular metabolism.

We mainly used the second option, hence the following equations will be given with that term, anyway the user can impose the constant consumption rate if desired.
Thus, we multiply (2.8) by $\alpha\psi_m^{-1}$ and, since V is arbitrary, we get

$$\frac{\partial}{\partial t}\mathrm{PO_2}^m = \psi_m^{-1}\frac{A_v^3}{V_t} P \left(\mathrm{PO_2}^3 - \mathrm{PO_2}^m \right) - \alpha\xi_0 \left(1 + \frac{\mathrm{PO_2}^{m,50}}{\mathrm{PO_2}^m} \right)^{-1} \tag{2.9}$$

The total amount of oxygen consumed by the muscle $\Lambda_{\mathrm{O_2}}$ [mol m^{-3} s^{-1} can be computed as

$$\Lambda_{\mathrm{O_2}}^{cons} = \frac{1}{T|\Omega|}\int_0^T \int_\Omega \psi_m \xi_0 \left(1 + \frac{\mathrm{PO_2}^{m,50}}{\mathrm{PO_2}^m} \right)^{-1} \tag{2.10}$$

2.1.3 Parameters description and calibration

In the model (2.6)-(2.9), the variables $\mathbf{u}_i$ and $\phi_{\alpha,\beta}$ are provided by the perfusion model (coupling discussed in section 2.1.4).
The compartment $i = 0$ corresponds to the arterial blood. We assume that in the latter compartment the chemical reactions are in equilibrium, that is

$$k_- \,\mathrm{SO_2}^i - k_+ \alpha^{-n} \left(1 - \mathrm{SO_2}^i \right) \left(\mathrm{PO_2}^i \right)^n = 0 \tag{2.11}$$

This corresponds to the Hill equation

$$\mathrm{SO_2}^i = \frac{\left(\mathrm{PO_2}^i\right)^n}{\left(\mathrm{PO_2}^i\right)^n + \left(\mathrm{PO_2}^{50}\right)^n} = \left(1 + \left(\frac{\mathrm{PO_2}^{50}}{\mathrm{PO_2}^i}\right)^n\right)^{-1} \tag{2.12}$$

where $\mathrm{PO_2}^{50} = \alpha \left(k_-/k_+\right)^{1/n}$ is the half-maximal effective oxygen partial pressure and where n is the cooperativity coefficient. In the following, we will assume that the arterial saturation $\mathrm{SO_2}^0 = \mathrm{SO_2}^a$ is given as a datum (constant in time), and that the oxygen partial pressure $\mathrm{PO_2}^0 = \mathrm{PO_2}^a$ is given by inverting (2.12).
It is possible to prove that, in the compartments $i = 1, 2$, saturation and partial oxygen pressure do not change with respect to arterial blood. Indeed, by setting $\mathrm{SO_2}^2 = \mathrm{SO_2}^1 = \mathrm{SO_2}^a$ and$\mathrm{PO_2}^2 = \mathrm{PO_2}^1 = \mathrm{PO_2}^a$ into (2.6), the equations are satisfied thanks to continuity equation. Therefore, we can focus ourselves in the third compartment only.
The Hill equation (2.12) allows to calibrate from the experimentally measured oxygen-hemoglobin dissociation curve the exponent n and the parameter $\mathrm{PO_2}^{50} = \alpha \left(k_-/k_+\right)^{1/n}$. Hence, for the dynamics of $\mathrm{SO_2}$, the only parameter left to be calibrated is k_-. Since the association-dissociation reaction between hemoglobin and oxygen is very fast (it could be considered as quasi-static) this could be set very large, without significantly affecting the solution.
Concerning the dynamics of $\mathrm{PO_2}$, we need to calibrate the following terms

- $\alpha\,[\mathrm{Hb}^*]$: both variables are available in the literature;
- $\frac{A_v^i}{V_t}P$: some variables could be found in the literature;
- ψ_i and $\psi_m = 1 - \sum_{i=1}^3 \psi_i$;
- $\alpha\xi_0$ and $\mathrm{PO_2}^{m,50}$: ruling oxygen consumption by the muscle.

Hence, to reduce the number of parameters to be calibrated, we denote the product $\tilde{P} = \frac{A_v^i}{V_t}P$ as the effective permeability (s^{-1}), and the product $\tilde{\xi_0} = \alpha\xi_0$ as the effective oxygen consumption rate [mmHg s^{-1}].
Reasonable values for the 7 parameters necessary for the model are collected in Tab.2.1.

2.1.4 Coupling with Perfusion model

We briefly recall the main features of the perfusion model of [3] (to which we refer for further details). The main unknowns in the compartments $i = 1, 2, 3$ are

- p_i blood pressure (Pa)
- $\mathbf{u_i}$ blood velocity (m s^{-1})

Variable	Measure Unit	Value
n	-	3.2
${PO_2}^{50}$	mmHg	27
α	mmHgm^3 mol^{-1}	722.6
[Hb*]	molm^{-3}	10
k_-	s^{-1}	100
$\tilde{P}$	s^{-1}	100
$\tilde{\xi}_0$	mmHg s^{-1}	40
${PO_2}^{m,50}$	mmHg	40
ψ_i	-	10^{-3}

Table 2.1: List of parameters.

Since the perfusion model of Section 2.1 is actually implemented with respect to the variable

$$\hat{\mathbf{u}}_i := \psi_i \mathbf{u}_i \tag{2.13}$$

the perfusion model is written as

$$\begin{aligned} \hat{\mathbf{u}}_i + \hat{K}_i \nabla p_i &= \mathbf{0} \quad \text{Darcy law} \\ \nabla \cdot \hat{\mathbf{u}}_i &= \hat{\phi}_{i-1,i} - \hat{\phi}_{i,i+1} \quad \text{Continuity equation} \end{aligned} \tag{2.14}$$

where

$$\begin{aligned} \hat{K}_i :=& \psi_i K_i, \quad \hat{\phi}_{i,j} := \psi_l \phi_{i,j}. \\ \hat{\phi}_{i,j} =& \beta_{i,j} \left(p_i - p_j \right) \end{aligned} \tag{2.15}$$

This helps with the calibration of the parameters. Indeed, the averaged flux of blood flowing from the compartment $i-1$ to i inside a tissue reference volume Ω and during a heart cycle $[0, T]$ can be computed as

$$\Psi_{CBF} := \frac{1}{T|\Omega|} \int_0^T \int_\Omega \rho_{blood}^{-1} \Phi_{i-1,i} = \frac{1}{T|\Omega|} \int_0^T \int_\Omega \psi_i \phi_{i-1,i} = \frac{1}{T|\Omega|} \int_0^T \int_\Omega \hat{\phi}_{i-1,i} \tag{2.16}$$

The quantity Ψ_{CBF} is known as coronary blood flow (CBF) and its typical values is known (nearly 0.8 min^{-1}, often expressed as 08 mL min^{-1} g^{-1}, as the weight of 1 mL of tissue is nearly 1 g).

In this manner, the complete model (2.6)-(2.9) can be rewritten as follows

$$
\begin{aligned}
\frac{\partial}{\partial t}\,\mathrm{PO_2}^i + \nabla\cdot(\mathrm{PO_2}^i\,\psi_i^{-1}\mathbf{\hat{u}_i}) &= n\alpha\,[\mathrm{Hb}*]\,k_-\left(\mathrm{SO_2}^i - (1-\mathrm{SO_2}^i)\left(\frac{\mathrm{PO_2}^i}{\mathrm{PO_2}^{50}}\right)^n\right)\\
&\quad + \psi_i^{-1}\hat{\phi}_{i-1,i}\,\mathrm{PO_2}^{i-1} - \psi_i^{-1}\hat{\phi}_{i,i+1}\,\mathrm{PO_2}^i\\
&\quad - \psi_i^{-1}\tilde{P}\left(\mathrm{PO_2}^i - \mathrm{PO_2}^m\right)\delta_{i3}\\
\frac{\partial}{\partial t}\,\mathrm{SO_2}^i + \nabla\cdot(\mathrm{SO_2}^i\,\psi_i^{-1}\mathbf{\hat{u}_i}) &= -k_-\left(\mathrm{SO_2}^i - (1-\mathrm{SO_2}^i)\left(\frac{\mathrm{PO_2}^i}{\mathrm{PO_2}^{50}}\right)^n\right)\\
&\quad + \psi_i^{-1}\hat{\phi}_{i-1,i}\,\mathrm{SO_2}^{i-1} - \psi_i^{-1}\hat{\phi}_{i,i+1}\,\mathrm{SO_2}^i\\
\frac{\partial}{\partial t}\,\mathrm{PO_2}^m &= \psi_m^{-1}\tilde{P}(\mathrm{PO_2}^3 - \mathrm{PO_2}^m) - \tilde{\xi}_0\left(1 + \frac{\mathrm{PO_2}^{m,50}}{\mathrm{PO_2}^m}\right)^{-1}
\end{aligned}
\tag{2.17}
$$

Then, the average flux of oxygen delivered and consumed can be computed as

$$
\Lambda_{\mathrm{O_2}}^{\mathrm{del}} = \frac{1}{T|\Omega|}\int_0^T\int_\Omega \tilde{P}\alpha^{-1}(\mathrm{PO_2}^3 - \mathrm{PO_2}^m) \tag{2.18}
$$

$$
\Lambda_{\mathrm{O_2}}^{\mathrm{cons}} = \frac{1}{T|\Omega|}\int_0^T\int_\Omega \psi_m\tilde{\xi}_0\alpha^{-1}\left(1 + \frac{\mathrm{PO_2}^{m,50}}{\mathrm{PO_2}^m}\right)^{-1} \tag{2.19}
$$

2.1.5 Reduced model

As each attached hemoglobin stores n molecules of oxygen, the concentration of oxygen stored into red blood cells is given by $n\,[\mathrm{Hb–O2}] = n\,[\mathrm{Hb}^*]\,\mathrm{SO_2}^i$. Conversely, the concentration of free oxygen is given by $[O_2]^i = \alpha^{-1}\,\mathrm{PO_2}^i$. Hence, we define the total oxygen concentration as

$$
[\mathrm{O_2^*}]^i := n\,[\mathrm{Hb}^*]\,\mathrm{SO_2}^i + \alpha^{-1}\,\mathrm{PO_2}^i
$$

From 2.17 we get

$$
\begin{aligned}
\frac{\partial}{\partial t}\,[\mathrm{O_2^*}]^i + \nabla\cdot([\mathrm{O_2^*}]^i\,\psi_i^{-1}\mathbf{\hat{u}}_i) &= \psi_i^{-1}\hat{\psi}_{i-1,i}\,[\mathrm{O_2^*}]^{i-1} - \psi_i^{-1}\hat{\psi}_{i,i+1}\,[\mathrm{O_2^*}]^i\\
&\quad - \psi_i^{-1}\tilde{P}\alpha^{-1}\left(\mathrm{PO_2}^i - \mathrm{PO_2}^m\right)\delta_{i3}
\end{aligned}
\tag{2.20}
$$

Let us take $i = 3$. We have $[\mathrm{O_2}]^2 = [\mathrm{O_2}]^a$ and $[\mathrm{O_2}]^3 \geqslant [\mathrm{O_2}]^m$. Integrating in the whole tissue domain Ω and over a full heart cycle $[0, T]$, since heartbeats are periodic and blood cannot flow outside the domain ($\mathbf{\hat{u}}_i \cdot \mathbf{n} = 0$ on $\partial\Omega$), by (2.18) and (2.16) we get an upper bound to the average oxygen flux

$$
\begin{aligned}
\Lambda_{0_2} &= \frac{1}{T|\Omega|}\int_0^T\int_\Omega \tilde{P}\alpha^{-1}\left(\mathrm{PO_2}^3 - \mathrm{PO_2}^m\right)\\
&\leqslant \Psi_{CBF}\left(n\,[\mathrm{Hb}^*]\,(\mathrm{SO_2}^a - \mathrm{SO_2}^m) + \alpha^{-1}\,(\mathrm{PO_2}^a - \mathrm{PO_2}^m)\right)
\end{aligned}
$$

An approximation of the model (2.6) - (2.9) can be obtained assuming that the chemical reaction between oxygen and hemoglobin is much faster than the other phenomena (diffusion through the membrane, convection), so that it can be considered at equilibrium. We will refer to this approximation as to a quasistatic-chemistry model and, in this case, the total oxygen concentration is given by a non linear, not invertible relation

$$[\mathrm{O}_2^*]^i = n\,[\mathrm{Hb}^*]\left(1+\left(\frac{{\mathrm{PO}_2}^{50}}{{\mathrm{PO}_2}^i}\right)^n\right)^{-1} + \alpha^{-1}\,{\mathrm{PO}_2}^i =: g({\mathrm{PO}_2}^i) \tag{2.21}$$

Hence, we obtain the reduced model

$$\begin{aligned} \frac{\partial}{\partial t}[\mathrm{O}_2^*]^i + \nabla\cdot\left([\mathrm{O}_2^*]^i\,\psi_i^{-1}\hat{\mathbf{u}}_i\right) &= \psi_i^{-1}\hat{\phi}_{i-1,i}\,[\mathrm{O}_2^*]^{i-1} - \psi_i^{-1}\hat{\phi}_{i,i+1}\,[\mathrm{O}_2^*]^i \\ &\quad - \psi_i^{-1}\tilde{P}\alpha^{-1}\left((g^{-1}([\mathrm{O}_2^*]^i) - {\mathrm{PO}_2}^m\right)\delta_{i3} \\ \frac{\partial}{\partial t}{\mathrm{PO}_2}^m &= \psi_m^{-1}\tilde{P}\left(g^{-1}([\mathrm{O}_2^*]^3) - {\mathrm{PO}_2}^m\right) \\ &\quad - \tilde{\xi}_0\left(1+\frac{{\mathrm{PO}_2}^{50}}{{\mathrm{PO}_2}^m}\right)^{-1} \end{aligned} \tag{2.22}$$

and the average flux of oxygen delivered (see (2.7)) can be computed as

$$\Lambda_{\mathrm{O}_2}^{\mathrm{del}} = \frac{1}{T|\Omega|}\int_0^T\int_\Omega \tilde{P}\alpha^{-1}(g^{-1}([\mathrm{O}_2^*]^3) - {\mathrm{PO}_2}^m) \tag{2.23}$$

Starting from the reduced model in the system of eq.(2.22), we want to compute the total oxygen concentration in the third compartment $[\mathrm{O}_2^*]^3$ and the oxygen pressure in the muscle ${\mathrm{PO}_2}^m$. We assume that the velocity of blood $\hat{\mathbf{u}}_3$, inward flux $\hat{\phi}_{2,3}$ from the second to the third compartment and outward flux from the last compartment to the venous system $\hat{\phi}_{3,4}$ are given. In case we are solving the model in standalone mode, i.e. without coupling with Perfusion model, taking as velocity and fluxes time dependent functions

$$\begin{aligned} \hat{\mathbf{u}}_3 &= \mathbf{0} \\ \hat{\phi}_{2,3} &= \Phi\sin\left(\pi\frac{t \bmod T_{HB}}{T_F}\right)^4 \mathbb{1}\left((t \bmod T_{HB}) < T_F\right) \\ \hat{\phi}_{3,4} &= \hat{\phi}_{3,\mathrm{veins}}(t) = \hat{\phi}_{2,3}(t) \end{aligned} \tag{2.24}$$

where $\Phi = 0.05$ s^{-1} is the flux magnitude, $T_{HB} = 0.8$ s is the period of a heart beat and $T_F = 0.6$ s is the duration of the simulated flux.

In case we are solving the model coupled with Perfusion model, we recover velocity and fluxes from [3] and 2.1.4 as

$$\begin{aligned}\hat{\mathbf{u}}_3 &= -K_3\nabla p_3\\ \hat{\phi}_{2,3} &= \beta_{23}(p_3 - p_2)\\ \hat{\phi}_{3,4} &= \hat{\phi}_{3,\text{veins}} = \gamma(p_3 - p_{\text{veins}})\end{aligned}\tag{2.25}$$

where K_3 is the permeability tensor of the third compartment, β_{23} and γ represent the inter-compartment pressure-coupling coefficients and p_{veins} is pressure that characterizes veins. Finally, in order to stabilize our problem and to simply some possible mathematical issues, it was added a diffusive part to the first equation in (2.22), hence it turns into

$$\begin{aligned}\frac{\partial}{\partial t}\left[\mathrm{O}_2^*\right]^i + \mu\triangle\left[\mathrm{O}_2^*\right]^i + \nabla\cdot\left(\left[\mathrm{O}_2^*\right]^i\psi_i^{-1}\hat{\mathbf{u}}_i\right) = \psi_i^{-1}\hat{\phi}_{i-1,i}\left[\mathrm{O}_2^*\right]^{i-1} - \psi_i^{-1}\hat{\phi}_{i,i+1}\left[\mathrm{O}_2^*\right]^i\\ -\psi_i^{-1}\tilde{P}\alpha^{-1}(g^{-1}(\left[\mathrm{O}_2^*\right]^i) - \mathrm{PO_2}^m)\delta_{i3}\end{aligned}\tag{2.26}$$

where μ is a not physical diffusive coefficient, tuned to have stable and realistic results.

2.2 Mathematical discretization

Regarding the numerical approximation of the problems, we have choosen to solve the 0D model, namely a 2-by-2 (nonlinear) system of ODEs, using a Runge-Kutta scheme of the 4th order, while we used the Finite Element Method to solve the 3D model which is a system of partial differential equations. We can find the details about discretization in the following sections.

2.2.1 Runge-Kutta method for the 0D model

The reduced 0D model in the third compartment, namely (2.22) rewritten for $i = 3$, is

$$\begin{aligned}\frac{d}{dt}[\mathrm{O}_2^*](t) &= \psi_3^{-1}\hat{\phi}_{2,3}\left[\mathrm{O}_2^*\right]^2(t) - \psi_3^{-1}\hat{\phi}_{3,\text{veins}}\left[\mathrm{O}_2^*\right]^3(t)\\ &\quad -\psi_3^{-1}\tilde{P}\alpha^{-1}(g^{-1}(\left[\mathrm{O}_2^*\right]^3(t)) - \mathrm{PO_2}^m(t)\\ \frac{d}{dt}\mathrm{PO_2}^m(t) &= \psi_m^{-1}\tilde{P}(g^{-1}(\left[\mathrm{O}_2^*\right]^3(t)) - \mathrm{PO_2}^m(t))\\ &\quad -\tilde{\xi}_0\left(1+\frac{\mathrm{PO_2}^{50}}{\mathrm{PO_2}^m(t)}\right)^{-1}\end{aligned}\tag{2.27}$$

In order to simplify the notation, we define

$$\begin{aligned}
y_1(t) &:= [\mathrm{O}_2^*]^3(t) \\
y_2(t) &:= \mathrm{PO_2}^m(t) \\
f_1(t, y_1(t), y_2(t)) &:= \psi_3^{-1}\hat{\phi}_{2,3}y_1(t) - \psi_3^{-1}\hat{\phi}_{3,\text{veins}}y_2(t) - \psi_3^{-1}\tilde{P}\alpha^{-1}\left(\left(g^{-1}y_1(t)\right) - y_2(t)\right) \\
f_2(t, y_1(t), y_2(t)) &:= \psi_m^{-1}\tilde{P}(g^{-1}(y_1(t)) - y_2(t)) - \tilde{\xi}_0\left(1 + \frac{\mathrm{PO_2}^{50}}{y_2(t)}\right)^{-1}
\end{aligned}$$

and $\mathbf{y}(t) = [y_1(t), y_2(t)]^T$, $\mathbf{f}(t, \mathbf{y}(t)) = [f_1(t, \mathbf{y}(t)), f_2(t, \mathbf{y}(t))]^T$, $\mathbf{y}_0 = [g(\mathrm{PO_2}^m), \mathrm{PO_2}^m]^T$ so that the problems becomes

$$\begin{cases}
\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t)) & t \in [0, T] \\
\mathbf{y}(0) = \mathbf{y}_0
\end{cases}$$

We fix a time step ΔT and discretize the time interval $[0, T]$ in $N_{\Delta T} := T/\Delta T$ time steps. We denote as $\mathbf{u}_n$ the approximation of $\mathbf{y}$ evaluated at t_n, namely $\mathbf{u}_n \approx \mathbf{y}(t_n)$, with $\{t_n\}_{n=0}^{N_{\Delta T}}$.
We can solve the ODEs system via Runge-Kutta explicit scheme of 4th order, which reads

$$\forall n = 1, \ldots, N_{\Delta t} \quad \mathbf{u}_{n+1} = \mathbf{u}_n + \frac{h}{6}\left(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4\right)$$

where

$$\begin{aligned}
\mathbf{K}_1 &:= \mathbf{f}\left(t_n, \mathbf{u}_n\right) \\
\mathbf{K}_2 &:= \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{u}_n + \frac{h}{2}\mathbf{K}_1\right) \\
\mathbf{K}_3 &:= \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{u}_n + \frac{h}{2}\mathbf{K}_2\right) \\
\mathbf{K}_4 &:= \mathbf{f}\left(t_{n+1}, \mathbf{u}_n + h\mathbf{K}_3\right)
\end{aligned}$$

2.2.2 Finite Element Method for the 3D model

Since $[\mathrm{O}_2^*]$ and $\mathrm{PO_2}^m$ are sufficiently regular in space and time, we can assume they belong to the space $V = L_2\left([0, T], H^1(\Omega)\right)$. The weak formulation of the first equation is obtained multiplying it by a generic function $v \in V$ and it reads

$$\begin{aligned}
\int_\Omega \frac{\partial}{\partial t}[\mathrm{O}_2^*]^3 v + \mu\triangle[\mathrm{O}_2^*]^3 v + \nabla\cdot([\mathrm{O}_2^*]^3\psi_3^{-1}\hat{\mathbf{u}}_3)v d\omega = \int_\Omega &\psi_3^{-1}\hat{\phi}_{2,3}[\mathrm{O}_2^*]^2 v - \psi_3^{-1}\hat{\phi}_{3,4}[\mathrm{O}_2^*]^3 v \\
&- \psi_3^{-1}\tilde{P}\alpha^{-1}(g^{-1}([\mathrm{O}_2^*]^3) - \mathrm{PO_2}^m)v d\omega
\end{aligned} \tag{2.28}$$

Integrating by parts the left hand side we get

$$\int_\Omega \frac{\partial}{\partial t}\left[\mathrm{O}_2^*\right]^3 v - \int_\Omega \mu\nabla\left[\mathrm{O}_2^*\right]^3 \nabla v + \int_{\partial\Omega} \mu\nabla\left[\mathrm{O}_2^*\right]^3 v\cdot\mathbf{n} - \int_\Omega \left[\mathrm{O}_2^*\right]^3 \psi_3^{-1}\hat{\mathbf{u}}_3\cdot\nabla v + \int_{\partial\Omega}\left[\mathrm{O}_2^*\right]^3 \psi_3^{-1}\hat{\mathbf{u}}_3\cdot\mathbf{n}v =$$
$$\int_\Omega \frac{\partial}{\partial t}\left[\mathrm{O}_2^*\right]^3 v - \int_\Omega \mu\nabla\left[\mathrm{O}_2^*\right]^3 \nabla v - \int_\Omega \left[\mathrm{O}_2^*\right]^3 \psi_3^{-1}\hat{\mathbf{u}}_3\cdot\nabla v$$

since blood and oxygen cannot flow outside the epicardium domain Ω (i.e. $\hat{\mathbf{u}}_3\cdot\mathbf{n} = 0$ and $\nabla\left[\mathrm{O}_2^*\right]^3\cdot\mathbf{n} = 0$ on $\partial\Omega$).

We take for the space discretization (both for oxygen concentration and muscle pressure) as finite dimensional subspace of $H^1(\Omega)$ the space of local linear polynomials in the space variable $X_h^1 = \left\{v_h \in C^0\left(\bar{\Omega}\right)(\Omega) : v_h|_K \in \mathbb{P}^1(K)\ \forall K \in \mathcal{T}_h\right\}$, where $\mathcal{T}_h$ is a proper triangulation of the space Ω. The semi-discrete Galerkin finite element formulation of (2.28) reads: find $\left(\left[\mathrm{O}_2^*\right]^3, \mathrm{PO_2}^m\right) \in X_h^1 \times X_h^1$ s.t.

$$\begin{aligned}
&\int_\Omega \frac{\partial}{\partial t}\left[\mathrm{O}_2^*\right]^3 v_h d\omega + a(\left[\mathrm{O}_2^*\right]^3, v_h) + b(\mathrm{PO_2}^m, v_h) = F(v_h)\ \forall\ v_h\ \in X_h \\
&\frac{\partial}{\partial t}\mathrm{PO_2}^m = \psi_m^{-1}\tilde{P}(g^{-1}(\left[\mathrm{O}_2^*\right]^3) - \mathrm{PO_2}^m) - \tilde{\xi}_0\left(1 + \frac{\mathrm{PO_2}^{50}}{\mathrm{PO_2}^m}\right)^{-1}
\end{aligned} \tag{2.29}$$

We introduce a basis $\{\varphi_i\}_{i=1}^{N_h}$ for the space X_h^1 and expand the unknowns $[\mathrm{O}_2^*]$ and $\mathrm{PO_2}^m$ in terms of the basis functions

$$\begin{aligned}
\left[\mathrm{O}_2^*\right]^3(x,y,z,t) &= \sum_{j=1}^{N_h}\left[\mathrm{O}_2^*\right]_j^3(t)\varphi_j(x,y,z) \\
\mathrm{PO_2}^m(x,y,z,t) &= \sum_{j=1}^{N_h}\mathrm{PO}_{2j}^{\ m}(t)\,\varphi_j(x,y,z)
\end{aligned} \tag{2.30}$$

Let us subdivide the time interval $(0,T]$ into $N_{\Delta t}$ sub-intervals of dimension Δt and fix the time instants $t_k = k\Delta t$ for $k = 0,\dots,N_{\Delta t}$. We decided to use a semi-implicit method for the time discretization scheme. In particular, we used an

- implicit treatment for the time derivative and the linear terms
- explicit treatment for the non linear terms (i.e. g function) and to decouple the two problems

Using the notation

$$\begin{aligned}
\left[\mathrm{O}_2^*\right]^{3,k} &= \left[\mathrm{O}_2^*\right]^3(x,y,z,t_k) \\
\mathrm{PO_2}^{m,k} &= \mathrm{PO_2}^m(x,y,z,t_k)
\end{aligned} \tag{2.31}$$

we get the two separated problems

$$\forall k = 0,...,N_{\Delta t} - 1 \text{ find } \left[\mathrm{O}_2^*\right]^{3,k+1} \in X_h^1,\ \mathrm{PO_2}^{m,k+1} \in X_h^1 \text{ s.t.}$$

$$
\begin{aligned}
\int_\Omega \frac{[\mathrm{O}_2^*]^{3,k+1}}{\Delta t} v_h d\omega - \int_\Omega \mu \nabla [\mathrm{O}_2^*]^{3,k+1} \nabla v_h d\omega \\
- \int_\Omega [\mathrm{O}_2^*]^{3,k+1} \psi_3^{-1} \hat{\mathbf{u}}_3 \cdot \nabla v_h d\omega + \int_\Omega \psi_3^{-1} \hat{\phi}_{3,4} [\mathrm{O}_2^*]^{3,k+1} v_h d\omega \\
= \int_\Omega \frac{[\mathrm{O}_2^*]^{3,k}}{\Delta t} v_h d\omega + \int_\Omega \psi_3^{-1} \hat{\phi}_{2,3} [\mathrm{O}_2^*]^{2,k+1} v_h d\omega \\
- \int_\Omega \psi_3^{-1} \tilde{P} \alpha^{-1} \left(g^{-1} \left([\mathrm{O}_2^*]^{3,k} \right) - \mathrm{PO_2}^{m,k} \right) v_h d\omega \quad \forall \ v_h \ \in X_h \\
\frac{\mathrm{PO_2}^{m,k+1}}{\Delta t} = \frac{\mathrm{PO_2}^{m,k}}{\Delta t} + \psi_m^{-1} \tilde{P} \left(g^{-1} \left([\mathrm{O}_2^*]^{3,k} \right) - \mathrm{PO_2}^{m,k} \right) - \tilde{\xi}_0 \left(1 + \frac{\mathrm{PO_2}^{50}}{\mathrm{PO_2}^{m,k}} \right)^{-1}
\end{aligned}
\tag{2.32}
$$

Substituting (2.30) in (2.32) and choosing $v_h = \varphi_i$ for any $i = 1, \ldots, N_h$ we get:

$$
A\mathrm{O_2}^{3,k+1} = \mathbf{F} \tag{2.33}
$$

where

$$
\begin{aligned}
\left([\mathrm{O}_2]^{3,k} \right)_i &= [0_2 *]_i^3 (t_k) \\
[A]_{i,j} &= \int_\Omega \frac{1}{\Delta t} \varphi_j \varphi_i d\omega - \int_\Omega \mu \nabla \varphi_j \nabla \varphi_i - \int_\Omega \varphi_j \psi_3^{-1} \hat{\mathbf{u}}_3 \cdot \nabla \varphi_i d\omega + \int_\Omega \psi_3^{-1} \hat{\phi}_{3,4} \varphi_j \varphi_i d\omega \\
[F]_i &= \int_\Omega \frac{[\mathrm{O}_2^*]^{3,k}}{\Delta t} \varphi_i d\omega + \int_\Omega \psi_3^{-1} \hat{\phi}_{2,3} [\mathrm{O}_2^*]^{2,k+1} \varphi_i d\omega - \int_\Omega \psi_3^{-1} \tilde{P} \alpha^{-1} \left(g^{-1} \left([\mathrm{O}_2^*]^{3,k} \right) - \mathrm{PO_2}^{m,k} \right) \varphi_i d\omega
\end{aligned}
$$

and

$$
\mathbf{PO}_2^{m,k+1} = \mathbf{PO}_2^{m,k} + \Delta t \psi_m^{-1} \tilde{P} \left(g^{-1} \left(\mathrm{O_2}^k \right) - \mathbf{PO}_2^{m,k} \right) - \tilde{\xi}_0 \left(1 + \frac{\mathrm{PO_2}^{50}}{\mathbf{PO}_2^{m,k}} \right)^{-1} \tag{2.34}
$$

2.2.3 Approximation of integral for oxygen fluxes computation

After solving the linear system (2.33) and muscle oxygen pressure (2.34), one may want to save the average solutions and then the average oxygen flux delivered and consumed, similar to those in (2.18), (2.19). In particular, having as time integration interval $[0, t]$ with t being the time of the simulation, the integrals are approximated using trapezoidal rule and neglecting the scaling on the duration of the simulation T as

$$\begin{aligned}
\mathrm{O_2}^{\mathrm{del}}(t) &= \frac{1}{|\Omega|}\int_0^t\int_\Omega \tilde{P}\alpha^{-1}(\mathrm{PO_2}^3 - \mathrm{PO_2}^m) = \int_0^t \tilde{P}\alpha^{-1}(\overline{\mathrm{PO_2}^3} - \overline{\mathrm{PO_2}^m}) \\
&\approx \sum_{t_i=0}^{t} \frac{1}{2\Delta t}\left\{\left[\tilde{P}\alpha^{-1}(\overline{PO^3}(t_i+\Delta t) - \overline{\mathrm{PO_2}^m}(t_i+\Delta t))\right]\right. \\
&\quad \left. + \left[\tilde{P}\alpha^{-1}(\overline{\mathrm{PO_2}^3}(t_i-\Delta t) - \overline{\mathrm{PO_2}^m}(t_i-\Delta t))\right]\right\} \\
\mathrm{O_2}^{\mathrm{cons}}(t) &= \frac{1}{|\Omega|}\int_0^t\int_\Omega \psi_m\tilde{\xi}_0\alpha^{-1}\left(1+\frac{\mathrm{PO_2}^{m,50}}{\mathrm{PO_2}^m}\right)^{-1} = \int_0^t \psi_m\alpha^{-1}\xi\left(\overline{\mathrm{PO_2}^m}(t)\right) \\
&\approx \sum_{t_i=0}^{t}\frac{1}{2\Delta t}\left[\psi_m\alpha^{-1}\xi\left(\overline{\mathrm{PO_2}^m}(t_i+\Delta t)\right) + \psi_m\alpha^{-1}\xi\left(\overline{\mathrm{PO_2}^m}(t_i-\Delta t)\right)\right]
\end{aligned}$$

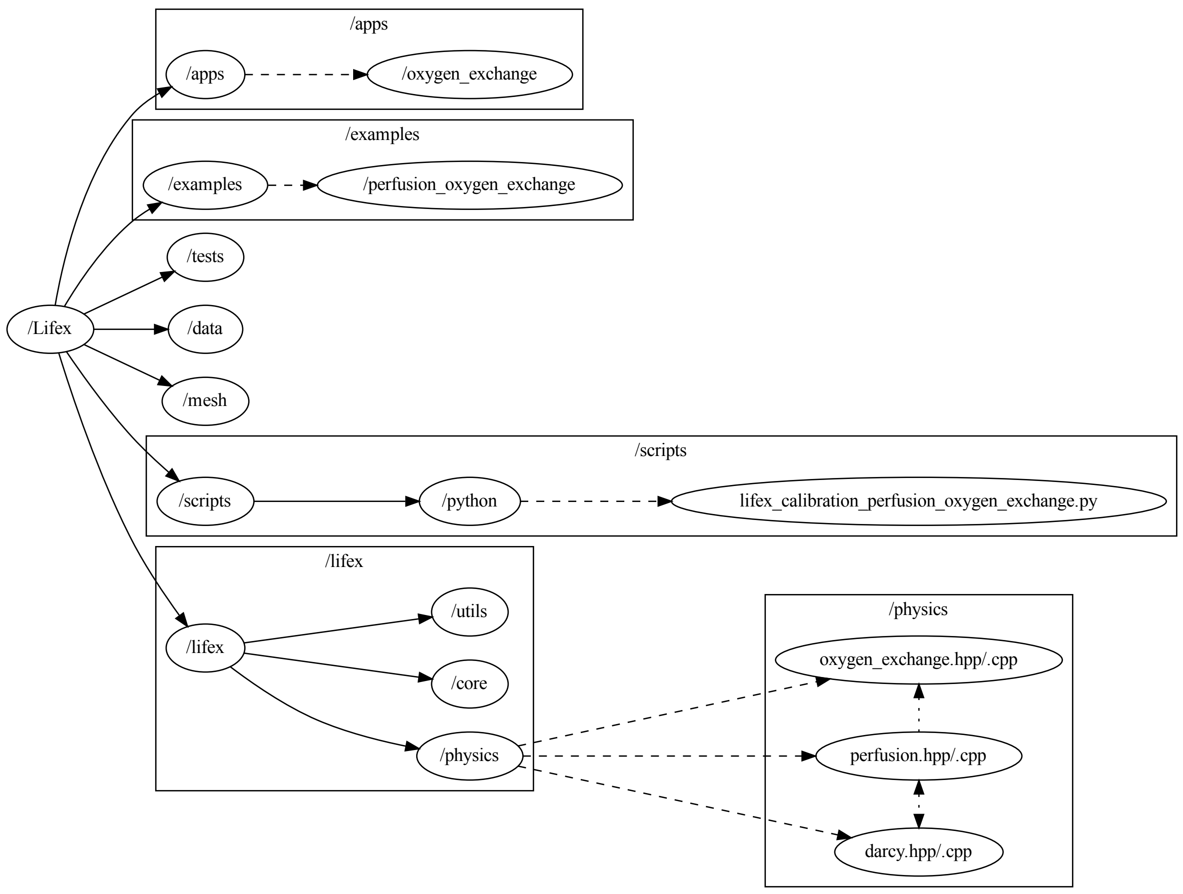
Chapter 3

Implementation

We implemented the OE model inside **life**^{x} library [7]. **life**^{x} is a high-performance Finite Element library mainly focused on mathematical models and numerical methods for cardiac applications. It is written in C++ and is based on the deal.II finite element core. It relies upon some dependencies, that can be found inside the `mk` package; to load the strictly necessary modules, use the command line:

```
module load gcc-glibc/11 dealii vtk
```

The main classes, with numerical tools and physical models, are implemented inside the directory `lifex/`, composed of three directories: `core/`, `physics/` and `utils/`. In particular, inside `physics/` are found all different classes modeling different cardiac functions. Those classes are used inside `apps/`, `tests/` and `examples/` of the different physics modelled, with additional information eventually taken from `data/` and `mesh/` directories.
In this section we describe the implementation of the `OxygenExchange` model class, its schematic structure, with a particular focus on the app `oxygen_exchange` which runs in standalone mode the OE dynamics, in 0d or in 3d mode, in contrast with the coupling example implemented in `examples/perfusion_oxygen_exchange`.



3.1 0D model

The 0D model can be solved through the execution of the app `oxygen_exchange`, setting properly some parameters specified in the next section, and it can solve the reduced model (2.17) or the complete model (2.22).

`OxygeExchange` constructor is written in the source file as follows:

```
#include "lifex/physics/oxygen_exchange.hpp"

#include <deal.II/dofs/dof_renumbering.h>

namespace lifex
{
  OxygenExchange::OxygenExchange(const std::string &subsection,
                                 const bool &       standalone_)
    : CoreModel(subsection)
    , standalone(standalone_)
    , triangulation(std::make_shared<utils::MeshHandler>(
        prm_subsection_path,
        mpi_comm,
        std::initializer_list<utils::MeshHandler::GeometryType>(
          {utils::MeshHandler::GeometryType::File,
           utils::MeshHandler::GeometryType::Hypercube}))
    , fe_scalar(1)
    , linear_solver(prm_subsection_path + " / Linear solver",
                    {"CG", "GMRES", "BiCGStab"},
                    "GMRES")
    , block_preconditioner(prm_subsection_path + " / Block preconditioner",
                           system_matrix)
    , csv_writer(mpi_rank == 0)
  {}
```

Listing 3.1: lifex/lifex/physics/oxygen_exchange/oxygen_exchange.cpp

Various of his component are necessary for the FE method.

```
  subsection Mesh and space discretization
    # Available options are: File | Hypercube.
    set Mesh type              = Hypercube # default: File

    # Specify whether the input mesh has hexahedral or tetrahedral elements↩
      .
    # Tetrahedral meshes can only be imported from file.
    # Available options are: Hex | Tet.
    set Element type           = Hex

    # Number of global mesh refinement steps applied to the initial grid.
    # Ignored if restart is enabled.
    set Number of refinements = 3          # default: 0

    # Degree of the pressure FE space.
    set FE space degree        = 1
```

Listing 3.2: build/apps/oxygen_exchange/lifex_oxygen_exchange.prm

The nonlinear system of ODEs (2.22) is solved every time step inside the `OxygenExchange::run_0d` member function through the call of the explicit Runge-Kutta method as follows

```
time = explicit_RK.evolve_one_time_step(
      [this](const double time, const Vector<double> &sol) {
        return evaluate_variations(time, sol);
      },
      time,
      prm_time_step,
      sol);
```

Listing 3.3: lifex/lifex/physics/oxygen_exchange/oxygen_exchange.cpp

which computes the solution at the next time step and updates the `time` member of the class. The inverse function of the nonlinear g function (2.21) is constructed by means of a lookup table, namely a map having as key $[O_2^*] = g(PO_2)$ and as value PO_2.

```
/// Set the g_inv map from [O_2*] to PO2.
    void
    set_g_inv_table()
    {
```

```
    for (double i = prm_initial_PO2; i <= prm_final_PO2; i = i + ↩
        prm_step_PO2)
      g_inv_table.insert(std::make_pair(g(i), i)); // <g(PO2), PO2>
  }
```

Listing 3.4: lifex/lifex/physics/oxygen_exchange/oxygen_exchange.hpp

When `g_inv(c_O2_star)` is called, it returns the linear interpolation between the value of the first element greater than or equal to the function argument and the values of the the previous one. The function has an undefined behaviour if the `lower_bound` of the argument is the first element of the map, as in the following code chunck.

```
  /// Function g.
  double
  g(const double &PO2) const
  {
    return prm_n_coop * prm_c_Hb_star * O2_Hb_diss_curve(PO2) +
           PO2 / prm_henry_constant_O2;
  }

  /// Inverse of g.
  double
  g_inv(const double &c_O2_star) const
  {
    auto   p  = g_inv_table.lower_bound(c_O2_star);
    double y2 = p->first;
    double x2 = p->second;
    // undefined behavior for prev if p = table.begin()
    double y1 = std::prev(p, 1)->first;
    double x1 = std::prev(p, 1)->second;
    return ((x2 - x1) / (y2 - y1)) * (c_O2_star - y1) + x1;
  }
```

Listing 3.5: lifex/lifex/physics/oxygen_exchange/oxygen_exchange.hpp

3.1.1 0D app

As every `life`^{x} application or example, to run `oxygen_exchange` app, is necessary to create an object `lifex::OxygenExchange` and specify some parameters that are required in order to run it. This is done through `declare_parameters` and `parse_parameters` member functions. It can be created the default `.prm` file using the command line

```
./lifex_oxygen_exchange -g
```

or set one specific file using `-f` option, followed by the name of the file. If no `-f` flaf is probided, a file named `executable_name.prm` is assumed to be available in the directory where the executable is run from.

In order to solve the 0D model, it is in particular necessary to set as `0D` the parameters `Model spatial dimension`. Moreover, the user can specify the desired model between com-

plete and reduced using the parameter `Enable complete model`. Other parameters allows the user to to choose between the Michaelis-Menten or constant muscle consumption dynamics, to build the inward/outward flux sinusoidal profile or to simulate a patient under physical activity or with a low blood oxygen saturation.
The parameters file is already filled with default values, hence the user can easily run a simulation of the 0d model only changing manually `Model spatial dimension`. The app then can be runned by calling the executable

```
./lifex_oxygen_exchange
```

where the user can also use the option `-o <PATH>` to specify in which folder the app must save the results. If the specified directory does not already exist, it will be created. By default, the current working directory is used. Absolute or relative paths can be specified for both the input parameter file and the output directory.
As other life^x apps, also `apps/oxygen_exchange` can be run also in parallel, using the `mpirun` or `mpiexec` wrapper commands, which may vary depending on the MPI implementation available on the machine. In particular, we can run the app by

```
mpirun -n <N_PROCS> ./lifex_oxygen_exchange [option...]
```

where `N_PROCS` is the desired number of parallel processes to run on. As a rule of thumb, 10000 to 100000 degrees of freedom per process should lead to the best performance.
In case of 0D app, the output consists in a `.csv` file containing the solutions of the model (SO_2, PO_2, $[O_2^*]$, ${PO_2}^m$) and others quantities of interest, like the average flux of oxygen O_2 delivered and consumed, computed as in (2.35) .
In validation phase, discussed in 4 chapter, we plotted all the results using `ParaView`, an open-source multi-platform data analysis and visualization application.

3.2 3D model

The solutions $[O_2^*]$ and ${PO_2}^m$ of (2.33) and (2.34) are collected into `LinAlg::MPI::BlockVector` data structures. In particular, we have 2 members for each solution: one containing the ghost entries and one (named appending `_owned` at the original name) without ghost entries. In parallel computations, vectors without ghost elements uniquely partition the vector elements between processors: each vector entry has exactly one processor that owns it, and this is the only processor that stores the value of this entry. On the other hand, ghosted vectors store some elements on each processor for which that processor is not the owner: you can read those

elements that the processor you are currently on stores but you cannot write into them because to make this work would require propagating the new value to all other processors that have a copy of this value. Since it is not possible to write into ghosted vectors, the only way to initialize this kind of vector is by assignment from a non-ghosted vector.
The `oxygen_exchange` app, when parameter `Model spatial dimension` is set `3D`, calls function `OxygenExchange::run()`, which sequentially calls the methods in order to assemble and solve the FE linear system for the oxygen concentration and then to update pressure.

```
void
void
OxygenExchange::run()
{
  if (prm_model_space_dim == "3D")
    run_3d();
  else // if (prm_model_space_dim == "0D")
    run_0d();
}

void
OxygenExchange::run_3d()
{
  initialize();
  setup_system();

  initialize_solution();

  compute_average_solutions();

  output_results();
  output_results_csv();
  csv_writer.write_line();

  while (time < prm_final_time)
    {
      time_advance();

      pcout << "Time step " << std::setw(6) << timestep_number
            << " at t = " << std::setw(8) << std::fixed
            << std::setprecision(6) << time;

      assemble();

      compute_average_O2_flux();

      solve_time_step();

      compute_average_solutions();
      compute_average_O2_flux();

      if (timestep_number % prm_output_every_n_timesteps == 0)
        {
          output_results();
          output_results_csv();
          csv_writer.write_line();
        }
    }
  }
```

Listing 3.6: lifex/lifex/physics/oxygen_exchange/oxygen_exchange.cpp

Firstly, through `initialize()` function is initialized the FE space `fe` and the quadrature formula `quadrature_formula`, using functions provided by `dealii` module, so that the system is set up properly. Then, at each time step, the FEM system matrix and right hand side of the system (2.33) is updated since it depends on previous $[O_2^*]$ and ${PO_2}^m$ solutions both with time-dependent inputs as blood velocity and inward/outward flux.
When `OxygenExchange` class is constructed, the linear solved is also initialized, in particular we used the `GMRES` solver is our work.
The `solve_time_step()` method finally solves (2.33) and finally the oxygen partial pressure of the muscle is updated in every degrees of freedom of the mesh, as specified in (2.34).

```
  void
  OxygenExchange::solve_time_step()
  {
    TimerOutput::Scope timer_section(timer_output,
                                     prm_subsection_path +
                                       " / Preconditioning and solving");

    block_preconditioner.initialize(system_matrix);

    // Solve [O2*]
    linear_solver.solve(system_matrix,
                        c_O2_star_owned,
                        system_rhs,
                        block_preconditioner);

    c_O2_star = c_O2_star_owned;

    // Update PO2 muscle
    for (auto idx : PO2_muscle.locally_owned_elements())
      {
        PO2_muscle_owned[idx] +=
          prm_time_step *
          PO2_muscle_variation(c_O2_star[idx], PO2_muscle_owned[idx]);
      }

    PO2_muscle_owned.compress(VectorOperation::add);
    PO2_muscle = PO2_muscle_owned;
  }
```

Listing 3.7: lifex/lifex/physics/oxygen_exchange/oxygen_exchange.cpp

3.2.1 3D app

Our app `apps/oxygen_exchange` allows also to solve the 3D uncoupled model, in which the coupling quantities are directly set instead of being computed from the Perfusion model. In particular, the velocitì is hard-coded to the zero vector field, while the inward/outward fluxes are constructed with a sinusoidal profiled that can be changed in the `.prm` file. Of course, in the `.prm` file the user must also specify `3D` as `Model spatial dimension`. We recall that the 3D model was implemented only in reduced version for computational efficiency, so in this case the parameter `Enable complete model` is ignored. On the other hand, in this case the user can find also the parameter related with space discretization, in order to be able to specify the kind of mesh (which can also be read from file), the number of mesh refinements and the degree of FE polynomials.

The 3D app can then be run as the 0D one, while the output of the simulations will contain a `XDMF` schema file (wrapped around a same named `HDF5` output file) that can be plot in `ParaView`. Also in this case, a `.csv` file is produced and it contains the same quantities of the 0D model, computed from the 3D simulations by a space average.

3.3 Coupled 3D model

In order to solve the Oxygen Exchange model coupled with the Perfusion model, we decided to add inside `Perfusion` class the protected attribute

```
  /// OxygenExchange object.
  std::shared_ptr<OxygenExchange> oxygen_exchange;
```

Listing 3.8: lifex/lifex/physics/perfusion.hpp

Moreover, we defined the `Perfusion` class as a `friend` class in `oxygen_exchange.hpp`, so that it has access to all protected and private members of `OxygenExchange`.

The user can choose if to solve the Oxygen Exchange model setting properly `prm_oxygen_exchange` of `Perfusion` class parameter. If it is true, then `Perfusion` constructs the `OxygenExchange` object passing `false` to the parameter `bool standalone`, which makes the OE model physically coupled with Perfusion one.

```
  if (prm_oxygen_exchange)
    {
      oxygen_exchange =
        std::make_shared<OxygenExchange>("Oxygen Exchange", false);
      oxygen_exchange->parse_parameters(params);
    }
```

Listing 3.9: lifex/lifex/physics/perfusion.cpp

In particular, `triangulation` member of OE is set with the one of DarcyAbstract thanks to `OxygenExchange::set_mesh_and_fe_space` method. Secondly, the inputs of the linear system are updated at every time step from the solution of the Perfusion problem (specifically from Darcy solutions) as specified in formula (2.25). This coupling is handled by

`Perfusion::update_oxygen_exchange` method which calls the getters of Darcy and the setters of OE to build up the desired objects.

```
void
  Perfusion::update_oxygen_exchange()
  {
    oxygen_exchange->set_flux_darcy(darcy->get_gamma(),
                                    darcy->get_p_veins(),
                                    darcy->get_beta_23(),
                                    darcy->get_p_owned());
    oxygen_exchange->set_p_3_owned(darcy->get_p_owned());
  }
```

Listing 3.10: lifex/lifex/physics/perfusion/perfusion.cpp

The first method update the block relative to the third compartment of `LinAlg::MPI::BlockVector inward_flux_darcy` and `LinAlg::MPI::BlockVector outward_flux_darcy` as in (2.25) as follows

```
void
    set_flux_darcy(const double &                    gamma,
                   const double &                    p_veins,
                   const double &                    beta_23,
                   const LinAlg::MPI::BlockVector &p_owned)
    {
      inward_flux_darcy_owned = 0.0;
      inward_flux_darcy_owned.block(2) += p_owned.block(1);
      inward_flux_darcy_owned.block(2) -= p_owned.block(2);
      inward_flux_darcy_owned.block(2) *= beta_23;

      outward_flux_darcy_owned = 0.0;
      outward_flux_darcy_owned.block(2) += p_owned.block(2);
      outward_flux_darcy_owned.block(2).add(-p_veins);
      outward_flux_darcy_owned.block(2) *= gamma;

      inward_flux_darcy_imposed_owned  = inward_flux(time);
      outward_flux_darcy_imposed_owned = outward_flux(time);

      inward_flux_darcy  = inward_flux_darcy_owned;
      outward_flux_darcy = outward_flux_darcy_owned;

      inward_flux_darcy_imposed  = inward_flux_darcy_imposed_owned;
      outward_flux_darcy_imposed = outward_flux_darcy_imposed_owned;
    }
```

Listing 3.11: lifex/lifex/physics/oxygen_exchange/oxygen_exchange.hpp

On the other hand, the second method simply extract the third block of pressure to compute the blood velocity. In order to do so, we pass through the member `std::unique_ptr<QuadratureFEMGradient> grad_p_3` which is initialize in `OxygenExchange::setup_system` in the following way

```
    grad_p_3 = std::make_unique<QuadratureFEMGradient>(p_3,
                                                        dof_handler_scalar,
                                              *quadrature_formula)↩
                                                ;
```
Listing 3.12: lifex/lifex/physics/oxygen_exchange/oxygen_exchange.cpp

Indeed, thanks to the data structure of the class `QuadratureFEMGradient` in `core/quadrature_evaluation`, we can easy evaluate the gradient in all the quadrature nodes during the assemble cycle, allowing us to compute the blood velocity.

3.3.1 Example of coupled model

As we have done for the 0D/3D apps, in order to run a simulation of the coupled model, we implemented the example `examples/perfusion_oxygen_exchange`. As for `examples/perfusion`, the user can specify the parameter `Coronaries model`, choosing how to model the fluid dynamics inside the coronaries, namely through the `FluidDynamics` class or `NetworkFlow` class. Since we were less interested in the coupling with coronaries model part in our simulations and we wanted to analyse the coupling between Darcy and OE models only, the fluid dynamics inside the coronaries is always modelled through `NetworkFlow`, with null source and deactivating Darcy contribute on fluid dynamics.
Here, we detailed some relevant modification to obtain this sort of complete decoupling between fluid dynamics and Darcy problem. Inside `perfusion_exchange.cpp` is constructed a `perfusion` member of type `Perfusion`, its inlet boundary conditions are initialized from the null pressure profile, whereas its outlet boundary conditions are initialized through `set_bcs` member function of `Perfusion` class.

```
perfusion.set_BCs(boundary_region_map,
                  inlet_bcs,
                  // Darcy Dirichlet BCs.
                  std::vector<utils::BC<utils::FunctionDirichlet>>{},
                  // Darcy No Neumann BCs means homogeneous Neumann ↩
                     BCs on
                  // the free tags.
                  std::vector<utils::BC<utils::FunctionNeumann>>{});
```
Listing 3.13: lifex/examples/perfusion_oxygen_exchange/perfusion_oxygen_exchange.cpp

Inside this function, outlet boundary conditions are defined using type `ScalarBC` and inside `perfusion.hpp` source file it is derived its specific type of scalar boundary conditions `BoundaryScalar`. In the other examples, in which the user may want to model also the coupling between fluid dynamics and Darcy, the outlet contribute to the coronaries is given by the average pressure computed from Darcy problem. To overcome this issue, we added parameter `prm_network_flow`, usually set `true` by default, here in our example set false so that `BoundaryScalar::value` method always return 0. Setting by default `prm_network_flow = true`, all the other examples already built inside **life**^{x} library still work.

```
/// Class in charge of computing outlet pressure for Network.
  class BoundaryScalar : public ScalarBC
  {
  public:
    /// Constructor.
    BoundaryScalar(const std::shared_ptr<DarcyAbstract> &darcy_,
                   const unsigned int                     region_,
                   const bool                             network_flow_ = ↩
                      true)
      : ScalarBC()
      , darcy(darcy_)
      , region(region_)
      , network_flow(network_flow_)
    {}

    /// Evaluation operator.
    virtual double
    value(const double & /*t*/)
    {
      if (network_flow)
        return darcy->compute_average_subdomain_pressure(
          region, 0); // Always on first compartment
      else
        return 0;
    };

  protected:
    const std::shared_ptr<DarcyAbstract>
                             darcy;  ///< Porous media model reference.
    const unsigned int region; ///< Perfusion region where this is ↩
       evaluated.
    const bool network_flow; ///< Boolean to tell if Network Flow is ↩
       simulated.
  };
```

Listing 3.14: lifex/lifex/physics/perfusion.hpp

One could specify manually the boundary map between `NetworkFlow` and `Perfusion` regions using the parameter `Use a coupling map`: in this case it must be provided the corresponding file `config/coupling_map.csv`. Also an inlet pressure can be provided using the parameter `Use a pressure profile` and filling the file `config/aortic_pressure_profile.csv`. In our example we decide to make the class use the default pressure profile (null) and default coupling map. In this way there is no inlet source for the fluid dynamics.

```
subsection Example perfusion OE
  # Use file aortic_pressure_profile.csv.
  set Use a pressure profile                = false

  # Use file coupling_map.csv.
  set Use a coupling map                    = false
```

Listing 3.15: build/examples/perfusion_oxygen_exchange/lifex_oxygen_exchange.prm

Regarding the solution of Darcy problem, it is possible to choose between the linear and nonlinear models using `Darcy model`. Moreover, in our example, the user can also specify an additional scalar source term as inlet flow for the Darcy problem, choosing between a uniformly scalar source, a spherical one and a sinusoidal inlet flow profile in the parameter `Inlet source type`.

```
subsection Scalar Source
  # Available options are: Uniform | Sphere | Sinusoidal.
  set Inlet source type = Sinusoidal
```

Listing 3.16: build/examples/perfusion_oxygen_exchange/lifex_oxygen_exchange.prm

Chapter 4

Tests and validation

In this section we can find the results of different kind of intermediete tests we implemented and performed to check the validity and integrity of the model and to debug our model implementation.

4.1 Comparison between complete/reduced 0D models

The first test we performed regarded the similarity of the complete and reduced model. Since the reduced model is build up under the quasistatic-chemistry assumption, namely that the chemical reaction of hemoglobin oxygenation happens almost instantly.

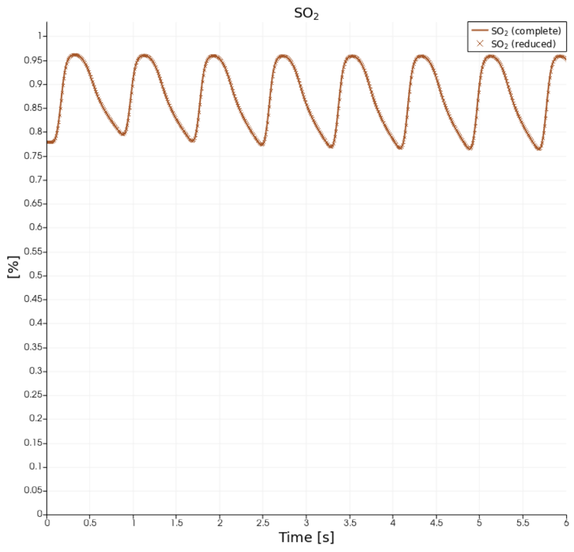
If this assumption would be right, we expect the two different models to provide almost identical results and, moreover, we expect the complete model to need a smaller time step w.r.t. to the complete one for stability reasons, since it has to properly capture the very fast chemical reaction.

To this end, we used the 0D app to perform two different simulation between $T_0 = 0$ s and $T = 6$ s with the same default parameters and initial condition and time discretization parameters. In the first one, we solved the complete models, while in the other one, we solved the reduced one. In both simulations, we computed in postprocessing the functions not directly solved in the model, namely $[O_2^*]$ for the complete model and PO_2 and SO_2 for the reduced one.

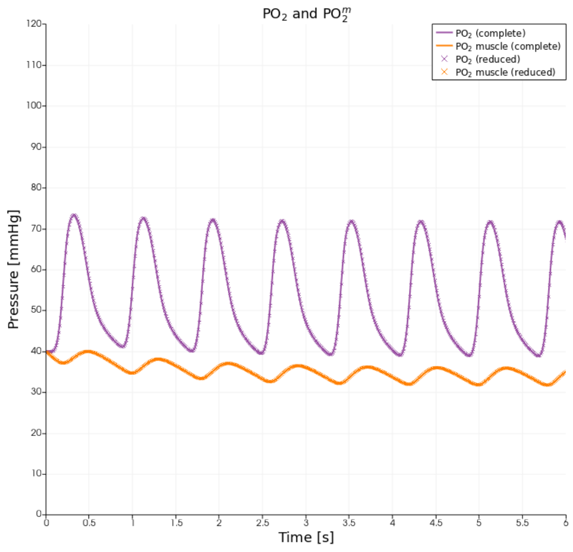
First of all, we noticed that using a time step of $\Delta t = 10^{-3}$ s, the complete model provides results which are very different w.r.t. ones of the reduced model.

Otherwise, using a time step of $\Delta t = 10^{-5}$ s the results are identical, confirming the validity of the quasistatic-chemistry assumption both with the fact that the complete model needs a very small timestep. Moreover, this test also confirms the consistency of our implementation of both the models.

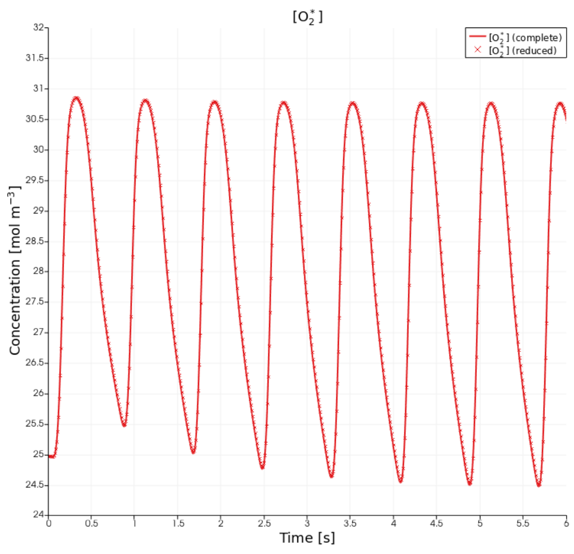
Thanks to that and since a so small time step would be unfeasible in a 3D simulation, we will



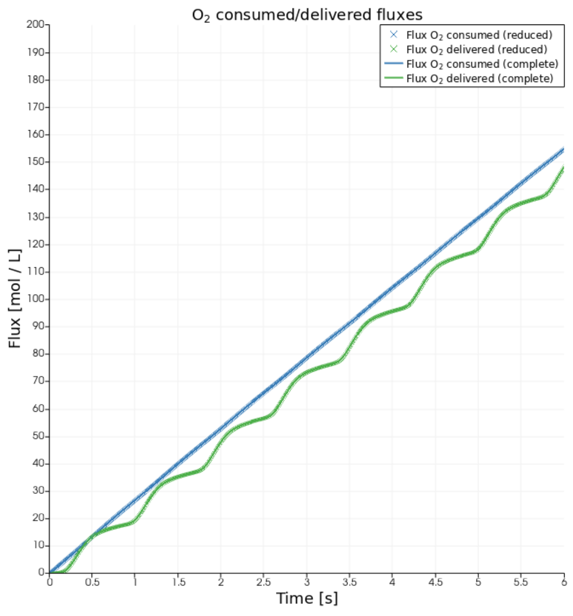
(a) SO_2.



(b) PO_2 and ${PO_2}^m$.



(c) $[O_2^*]$.



(d) O_2 consumed and delivered fluxes.

Figure 4.1: Comparison between complete and reduced 0D models with $\Delta t = 10^{-5}$ and $\Delta t = 10^{-2}$ respectively.

consider only the reduced model from now on.

4.2 Comparison between reduced 3D and 0D models

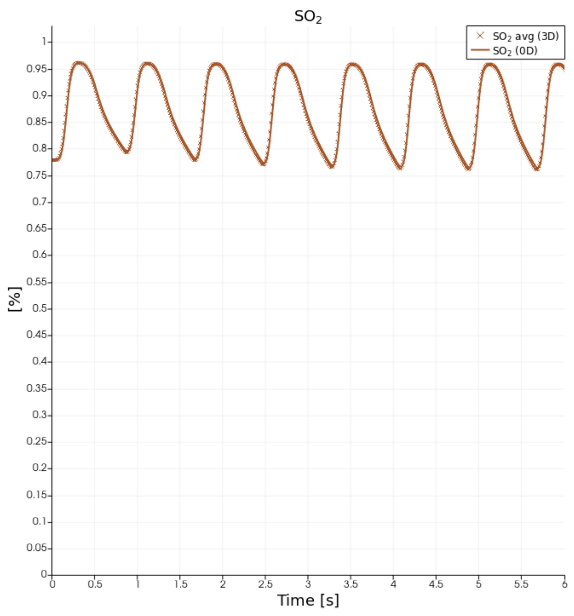
Another goal of our work, was to validate the implementation of the 3D model before passing to the coupling of other physics.
The 3D model was implemented setting null blood velocity and imposing the blood inward/outward flux point-wise in the mesh grid. Although the presence of a diffusive term, with a small coefficient only for regularization purposes, this means that we can compare the result of the 0D model with the results of 3D model evaluated in an arbitrary d.o.f. of the volume.
To this end, we used the 0D and 3D apps to run two different simulations with the same same parameters.
As expected, the solutions of 3D model is almost homogeneous in space, and their evolution in time in each mesh node is the same of the 0D case. This confirms that our implementation of the 3D model through FEM is correct.

4.3 Validation of coupled 3D model

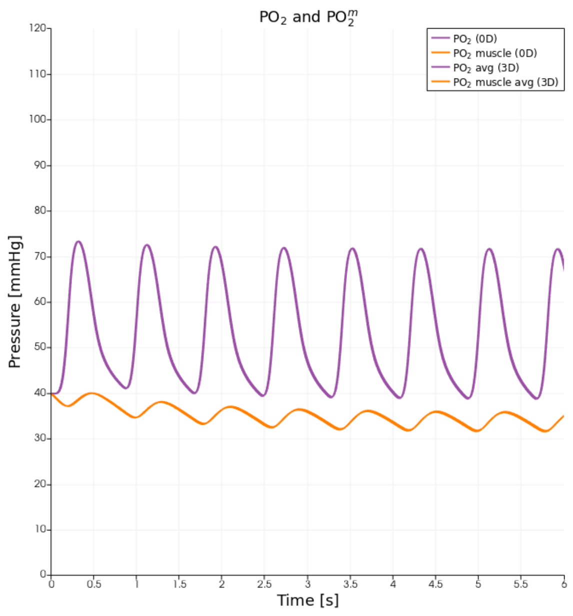
In this section we will show some tests we implemented and performed to validate the coupling between the OE model and the perfusion problem. Firstly, we will show how we checked that the initial flux would be properly propagated through the 3 compartments and then passed as input/output to OE model. Secondly, we will described the results of the comparison between the coupled model and the 3D one when same conditions where applied to both of them.

4.3.1 Calibration of Darcy source

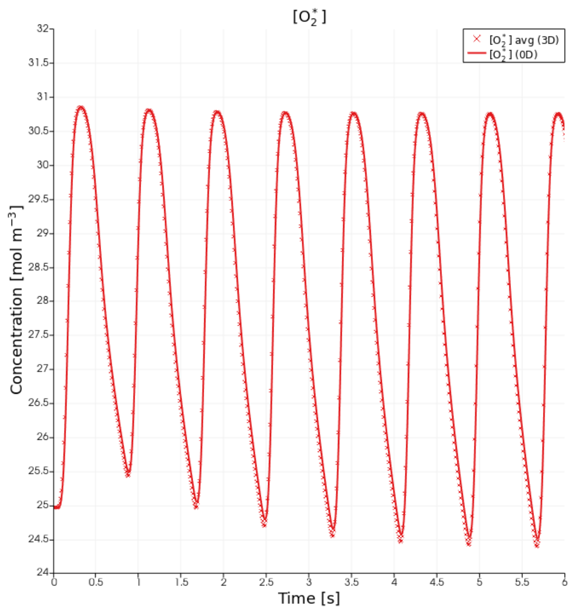
In order to verify the correctness of Darcy model implementation and in particular our use of that class, we replicated the results in section 4 (first test) of [3].
This test was performed in an ideal cubic domain imposing the parameters in the table and a inlet flow rate with a sinusoidal profile in time. Regarding the results, the authors verified that the outlet flux profile was in excellent agreement, showing that the mass conservation is respected, and moreover they observed a value of $\Delta p_{1,2} = 24.66$ mmHg.
Our aim, was to change the inlet flow as a constant flow uniformly distributed in a shpere having center in one vertex of the cube and radius equal to half of the cube length. To obtain the same results of the reference work, we proceed inversely, calibrating the value of the point-wise inlet flux starting from the previous parameters and $\Delta p_{1,2}$. This is done in the Python script `config/calibration.py` in this code chunck



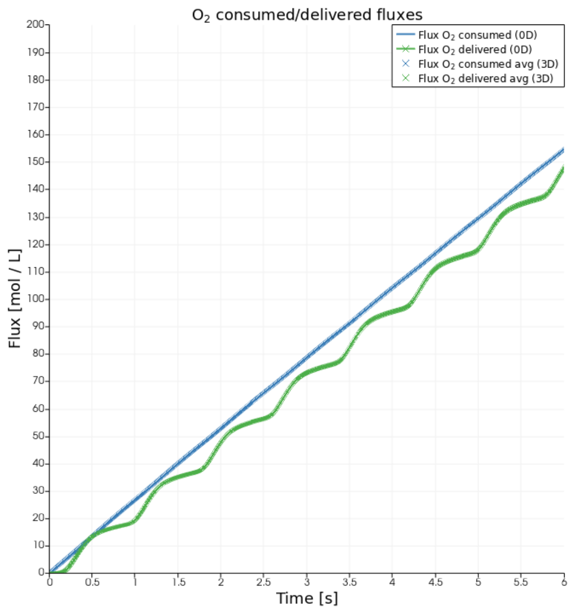
(a) SO_2.



(b) PO_2 and ${PO_2}^m$.



(c) $[O_2^*]$.



(d) O_2 consumed and delivered fluxes.

Figure 4.2: Comparison between 3D (average solutions) and 0D models with $\Delta t = 10^{-2}$.

Variable	Measure Unit	Value
K_3	$cm^2(Pa \cdot s)^{-1}$	10^{-3}
$\beta_{2,3}$	$(Pa \cdot s)^{-1}$	$3.62 \cdot 10^{-6}$
γ	$(Pa \cdot s)^{-1}$	10^{-4}
p_{veins}	Pa	2992.5

Table 4.1: List of parameters.

```python
import numpy as np
import pandas as pd
from prettytable import PrettyTable

# Parameters
beta_12 = 2.6e-6
beta_23 = 3.62e-6
gamma = 1e-4
p_veins = 22.5 * 133 # (from mmHg to Pa)
sphere_r = 5e-3
cube_l = 1e-2
sphere_vol = 1/8 * 4/3 * np.pi * sphere_r**3
cube_vol = cube_l**3

# Compute inlet source value
delta_p_12 = 24.66 * 133 # (from mmHg to Pa)
phi_01 = delta_p_12 * beta_12
value = phi_01 / (sphere_vol / cube_vol)

# Results
df = pd.read_csv('./results/perfusion.csv')
p_perf = np.around(np.array([df["p0_avg"][1], df["p1_avg"][1], df["p2_avg"]↩
    ][1]]), 2)
p = np.array([p_veins + gamma**(-1) * phi_01,
     p_veins + gamma**(-1) * phi_01 + beta_23**(-1) * phi_01,
     p_veins + gamma**(-1) * phi_01 + beta_23**(-1) * phi_01 + beta_12↩
        **(-1) * phi_01])
p = np.around(p, 2)
delta_p = np.around((p_perf - p) / p_perf, 2)

# Print results
print("Inlet source: {:4f}".format(value))
t = PrettyTable(['', 'Compartment 1', 'Compartment 2', 'Compartment 3'])
t.add_row(["Avg pressure [Pa]", p[0], p[1], p[2]])
t.add_row(["Perfusion pressure [Pa]", p_perf[0], p_perf[1], p_perf[2]])
t.add_row([ "Delta [%]", delta_p[0], delta_p[1], delta_p[2]])
print(t)
```

Listing 4.1: /examples/perfusion_oxygen_exchange/config/calibration.py

that we can run from `examples/perfusion_oxygen_exchange/` by

```
python ./../../../scripts/python/lifex_calibration_perfusion_↩
    oxygen_exchange.py
```

It provides a value of point-wise inlet flow of 0.130290 [s^{-1}].

Moreover, after the simulation is done, the script can be run again to compare the average pressures with ones analytically computed, providing this results

```
+------------------------+--------------+--------------+---------------+
|                        | Compartment 1 | Compartment 2 | Compartment 3 |
+------------------------+--------------+--------------+---------------+
|    Avg pressure [Pa]    |   3077.77    |   5433.42    |    8713.2     |
| Perfusion pressure [Pa] |   8981.75    |    5548.0    |   3081.78     |
|        Delta [%]        |     0.66     |     0.02     |    -1.83      |
+------------------------+--------------+--------------+---------------+
```

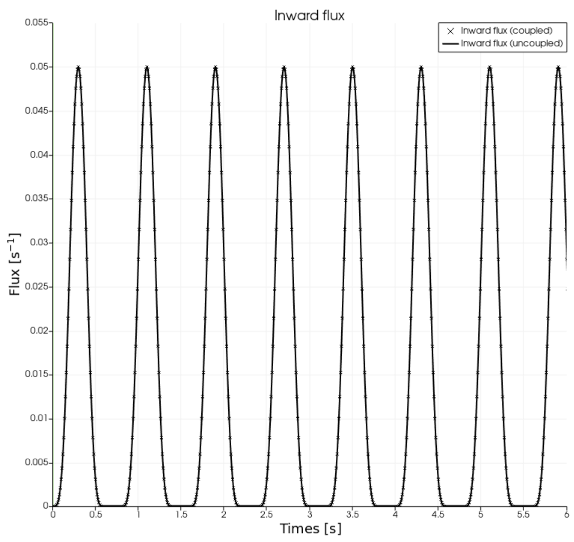
We can see that the percentage difference between simulated average pressures and analytical ones are very small, confirming that the model properly works also with different kind of inflow fluxes, as the one we imposed.

4.3.2 Comparison between coupled and 3D (uncoupled) models

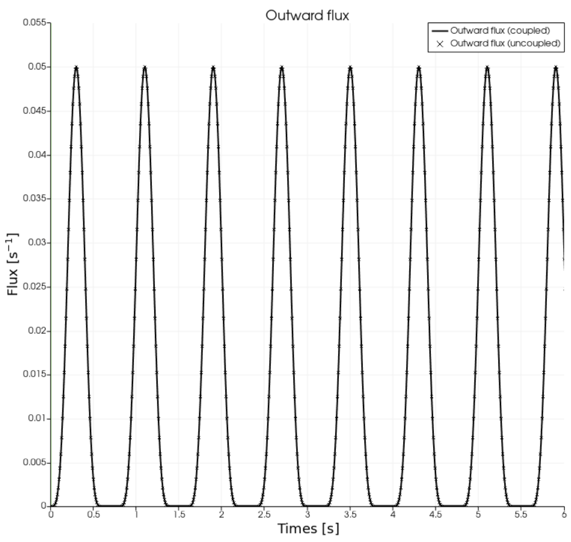
As a last step of our coupled model validation, we compared it to the 3D uncoupled one. We recall that the quantities involved in the coupling between OE and Perfusion were the blood velocity and the inward/outward fluxes. Of course, in order to our comparison to make sense, we need to have these quantities equal between the to models.
Hence, we imposed `apps/oxygen_exchange` zero blood velocity and sinusoidal flux profile (as in previous sections), while, in `examples/perfusion_oxygen_exchange`, we imposed as scalar source the same of `apps/oxygen_exchange`, constant in space. Thus, considering, problem (2.22) with $i = 3$ and using (2.25), we get the same quantities of (2.24), namely $\hat{\mathbf{u}}_3 = \mathbf{0}$ (since the pressures are spatially homogeneous in each compartment) and $\hat{\phi}_{2,3} = \hat{\phi}_{3,veins}$ equal to $\hat{\phi}_{0,1}$ scalar source, therefore to inward/outward fluxes imposed in the uncoupled model.
Firstly, we run a simulation of both the coupled and the uncoupled models on a cubic volume of $l = 10^{-1}$ m with $\Delta t = 10^{-2}$ s.
As you can see in Figure 4.3, the inward and outward fluxes in coupled case are equal to ones of the uncoupled case, confirming that the fluxes are correctly computed from Perfusion. Noticing that also the blood velocity is almost zero also in the coupled case, we can expect the two solutions to be indistinguishable.
As a matter of fact, in Figure 4.4 we can see that solved quantities both with the quantities of interest are identical in the two cases so that our implementation of the coupled model can be considered to be correct.

Smaller volume

Since the heart has a volume in the order of cm^3, we run a simulation with a smaller length $l = 10^{-2}$ m, again with $\Delta t = 10^{-2}$ s.



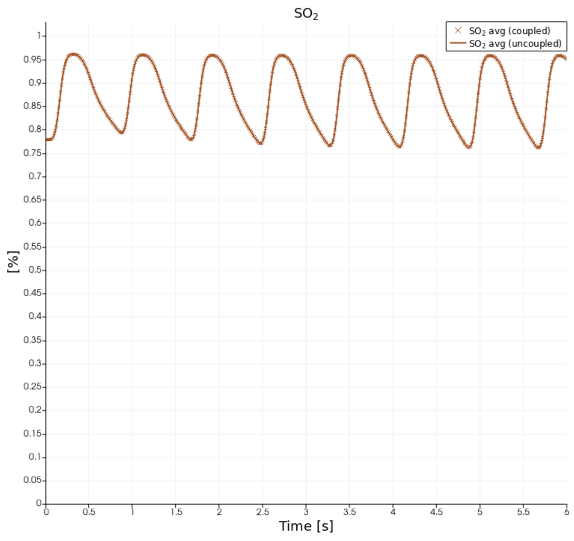
(a) Inward fluxes.



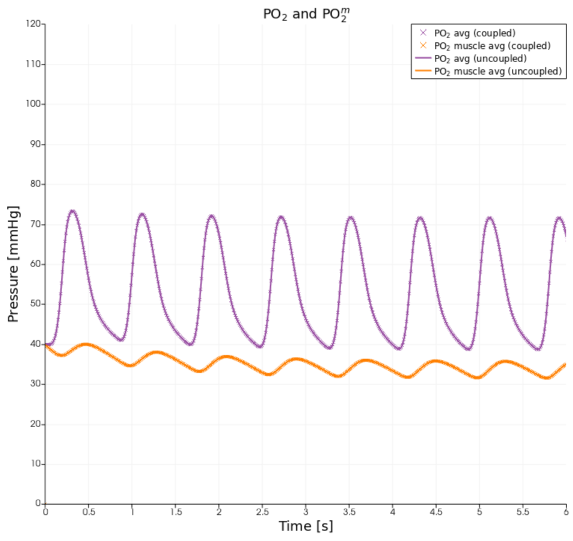
(b) Outward fluxes.

Figure 4.3: Inward and outward fluxes for coupled and uncoupled model on a cube of length $l = 10^-1$ m and with $\Delta t = 10^{-2}$ s.

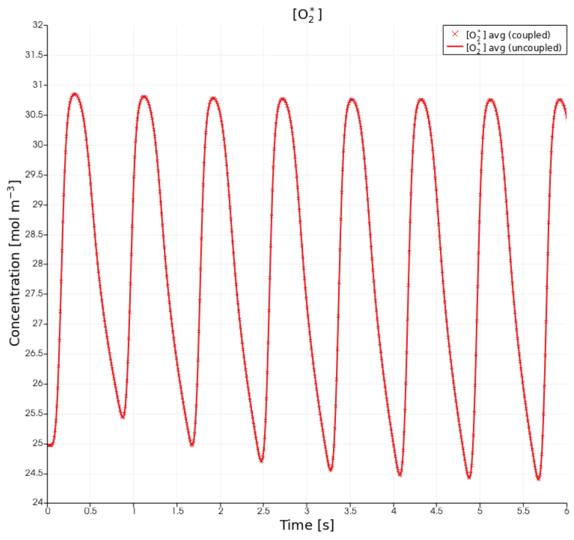
As you can see in Figure 4.5, this time the inward and outward fluxes in coupled case differs consistently from the same one of the uncoupled case. In particular, notice that the spike is higher and that the fluxes are never numerically zero as the uncoupled ones.
Of course, this has a non negligible influence on the solutions of the coupled model that differs consistently from the ones of the uncoupled one, as shown in Figure 4.6.
In order to verify if this is related to the time-approximation error, we simulated also the with the same parameters but $\Delta t = 10^{-4}$ s. As we can see in Figure 4.7 the solutions becomes more similar for a smaller Δt, hence we can consider our test to be successful and to confirm our hypothesis.



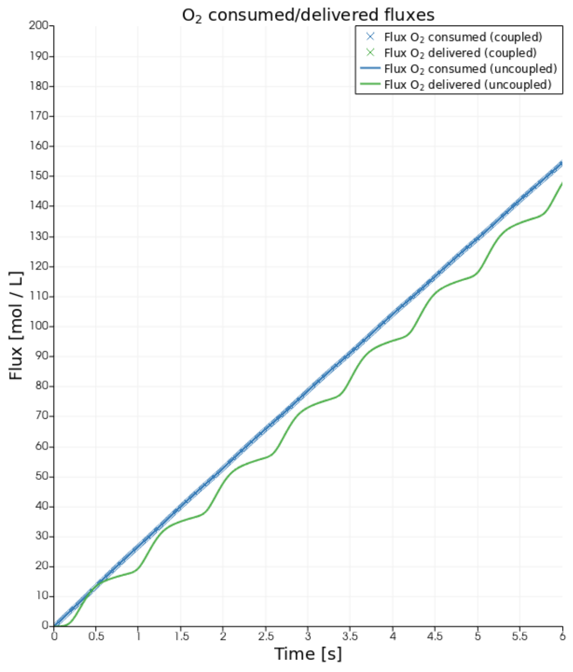
(a) SO_2.



(b) PO_2 and ${PO_2}^m$.

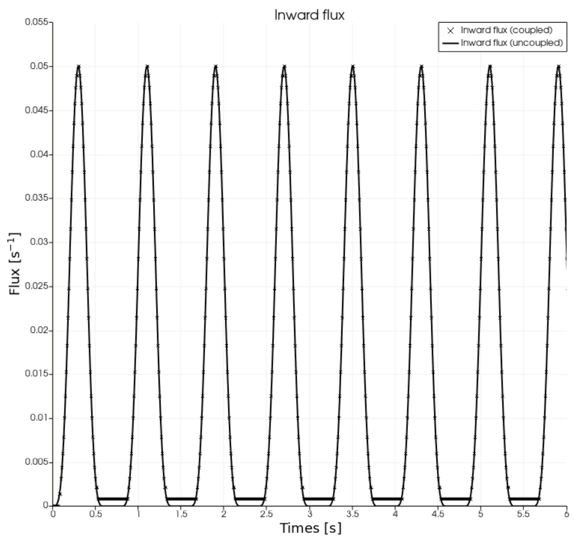


(c) $[O_2^*]$.

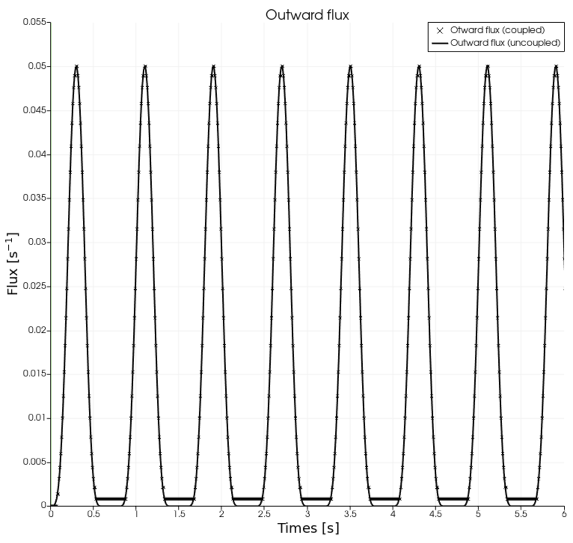


(d) O_2 consumed and delivered fluxes.

Figure 4.4: Comparison between average solutions of coupled and uncoupled model on a cube of length $l = 10^-1$ m and with $\Delta t = 10^{-2}$ s.

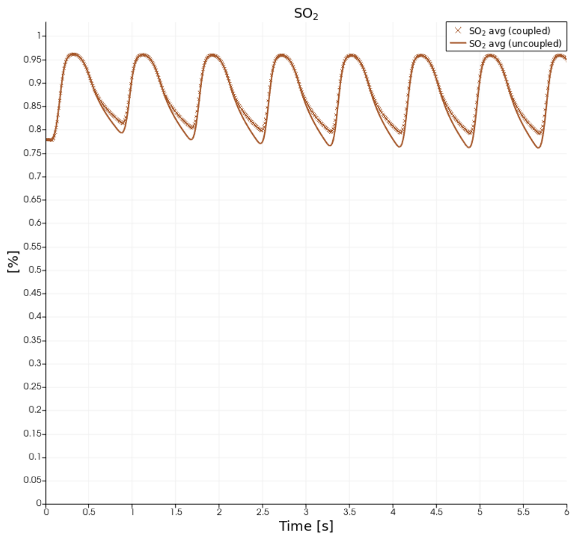


(a) Inward fluxes.

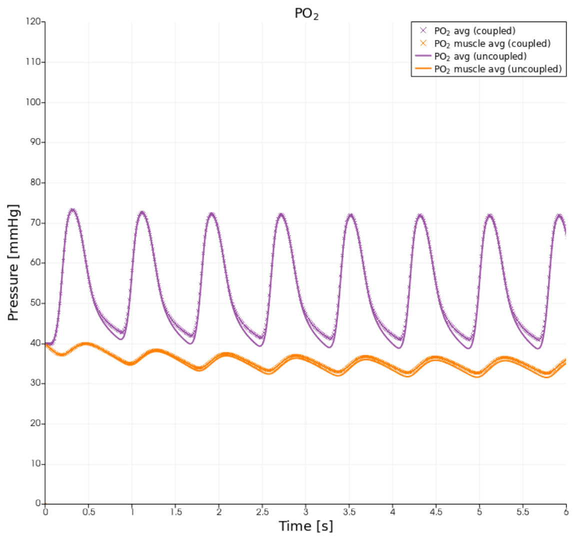


(b) Outward fluxes.

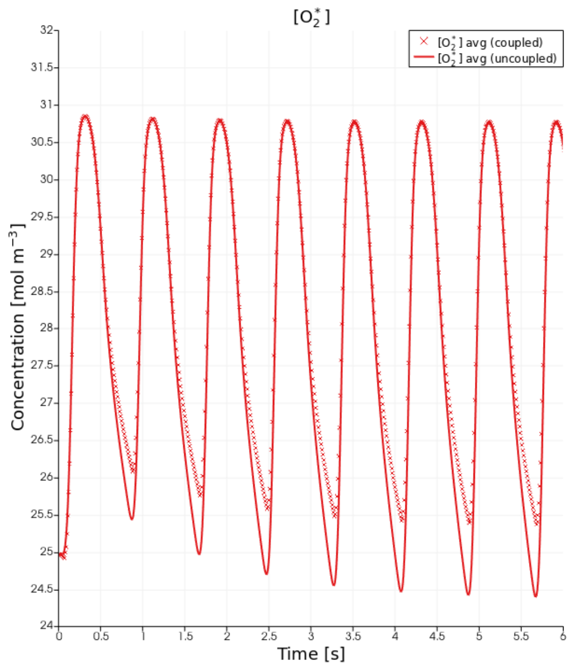
Figure 4.5: Inward and outward fluxes for coupled and uncoupled model on a cube of length $l = 10^{-2}$ m and with $\Delta t = 10^{-1}$ s.



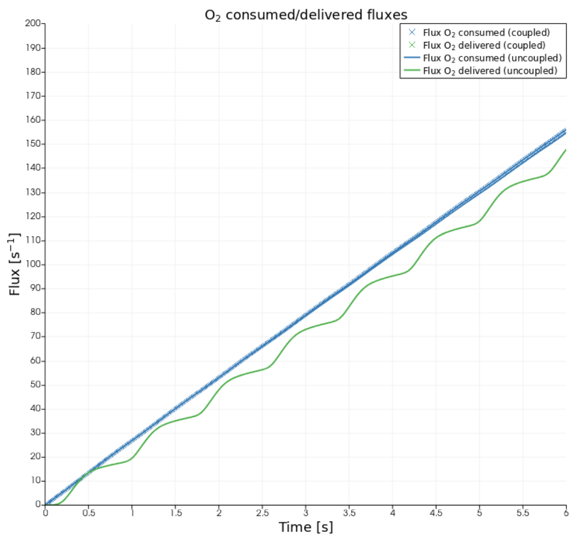
(a) SO$_2$.



(b) PO$_2$ and PO$_2{}^m$.

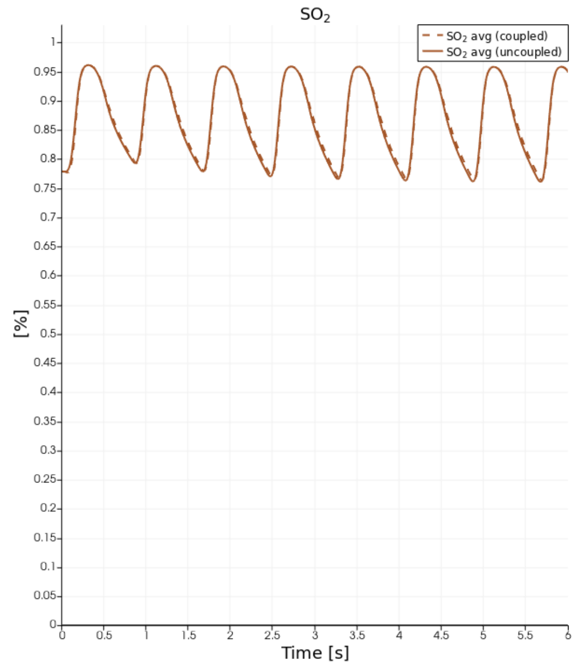


(c) [O$_2^*$].

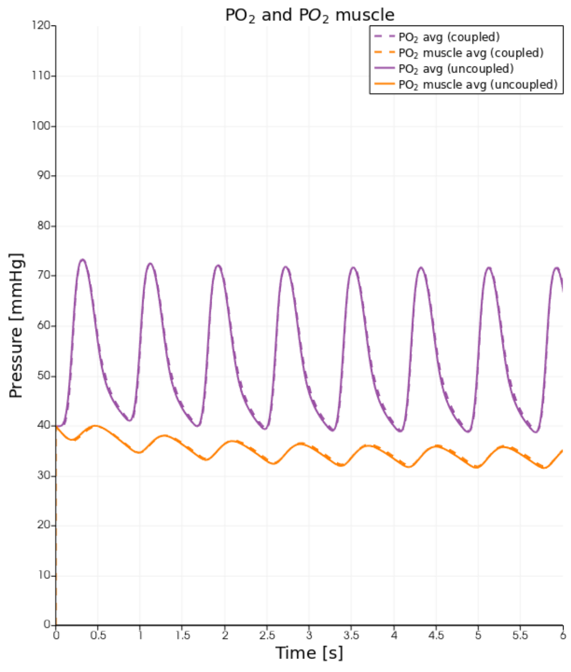


(d) O$_2$ consumed and delivered fluxes.

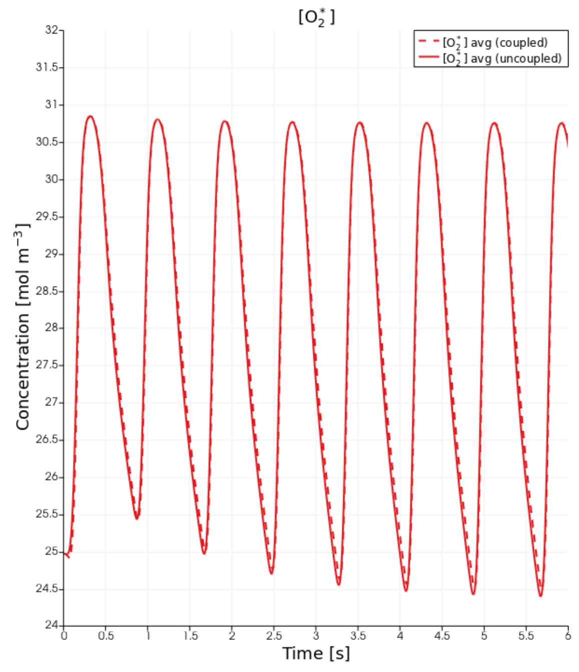
Figure 4.6: Comparison between average solutions of coupled and uncoupled model on a cube of length $l = 10^{-2}$ m and with $\Delta t = 10^{-1}$ s.



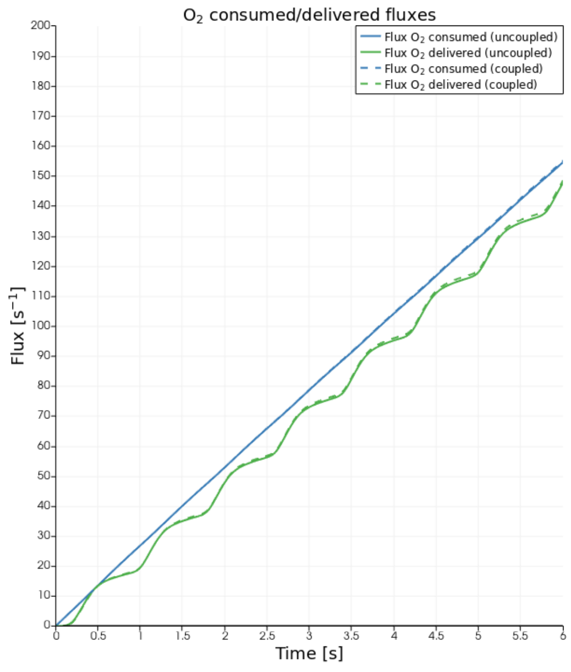
(a) SO_2.



(b) PO_2 and ${PO_2}^m$.



(c) $[O_2^*]$.



(d) O_2 consumed and delivered fluxes.

Figure 4.7: Comparison between average solutions of coupled and uncoupled model on a cube of length $l = 10^{-2}$ m and with $\Delta t = 10^{-4}$ s.

Chapter 5

Results

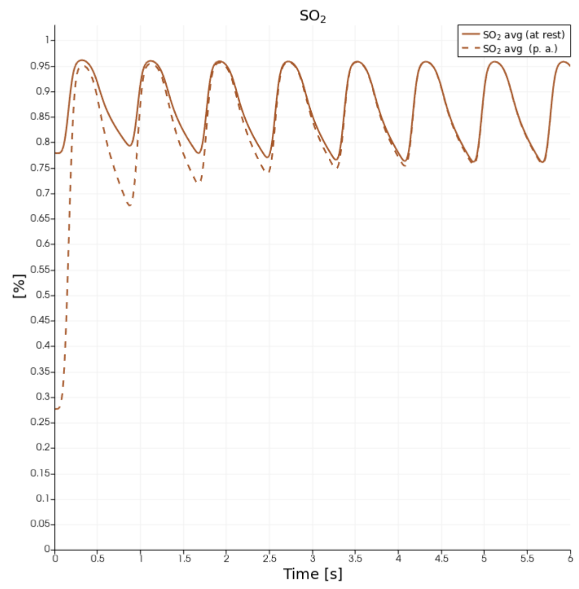
After having validating the model and our implementation, we used it to simulate some real life situations, in particular we studied the difference between a patient at rest and under physical activity, possibly in combination with infection by COVID-19 disease.

5.1 Effect of physical activity

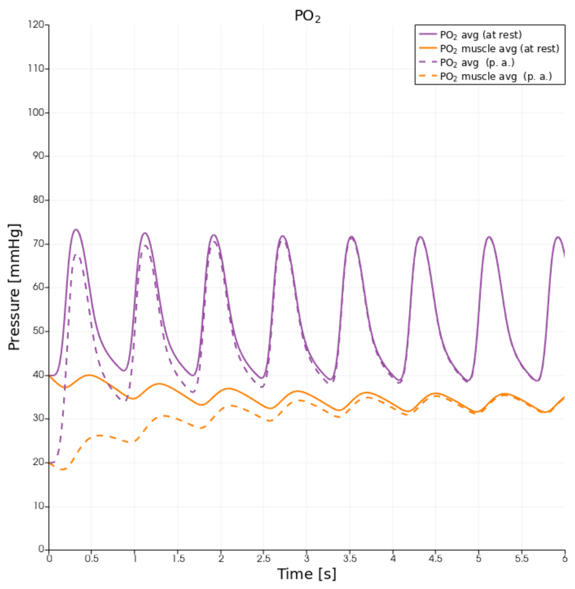
We started studying the influence of physical activity, i.e. walking or running, on the parameters of the patient. In order to do this, we simulated two cases with the default parameters we used also before, but changing the muscle dynamics maximum effective consumption rate: $\tilde{\xi}_0 = 40$ mmHg s^{-1} for a patient at rest and $\tilde{\xi}_0 = 20$ mmHg s^{-1} for a patient under physical activity.
As we can see the results in Figure 5.1, an increasing in the oxygen consumption by the cardiac muscle affects the model output: in particular we notice a reduction of SO_2, PO_2 and $[O_2^*]$ caused by hypotension affecting tissue and organs. Moreover, we notice that we have an increasing in oxygen flux delivered from the capillaries to the muscle, which means that, as we expect, the heart asks for more oxygen to be able to work properly. Anyway, if the oxygen consumption increases too much, the only oxygen delivered quantity may be even not enough to produce ATP, hence the muscolar tissue may use its own oxygen amount. This reflects a condition in which the muscles ask for more oxygen, as in a patient under physical activity.

5.2 Effect of COVID-19 disease

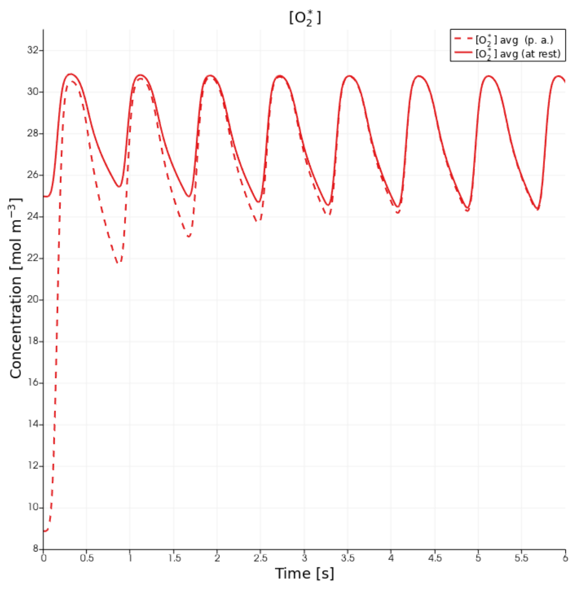
Moreover, we studied the effect of COVID-19 disease on a patient (at rest) through the model we implemented. Hence, we run two simulations with parameters found in literature for physiological and pathological conditions as in the Table 5.1.
The value of ${SO_2}^a$ is the minimum saturation level which has to be reached providing mechanical



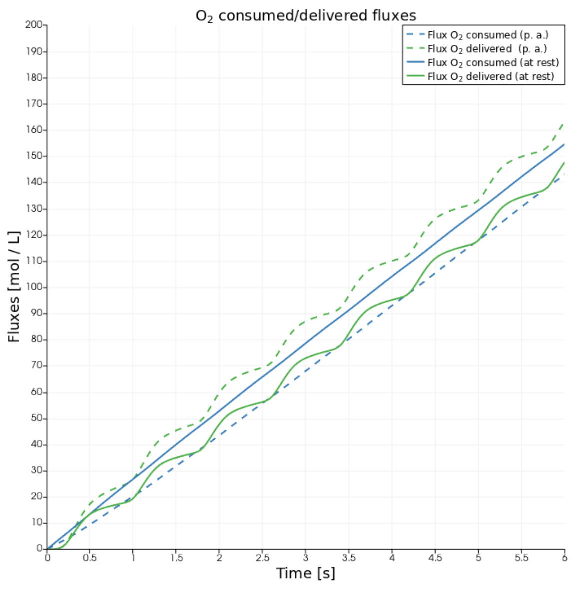
(a) SO_2.



(b) PO_2 and $PO_2{}^m$.



(c) $[O_2^*]$.



(d) O_2 consumed and delivered fluxes.

Figure 5.1: Comparison between patient at rest and one under physical activity, $l = 10^{-1}$ m and with $\Delta t = 10^{-2}$ s.

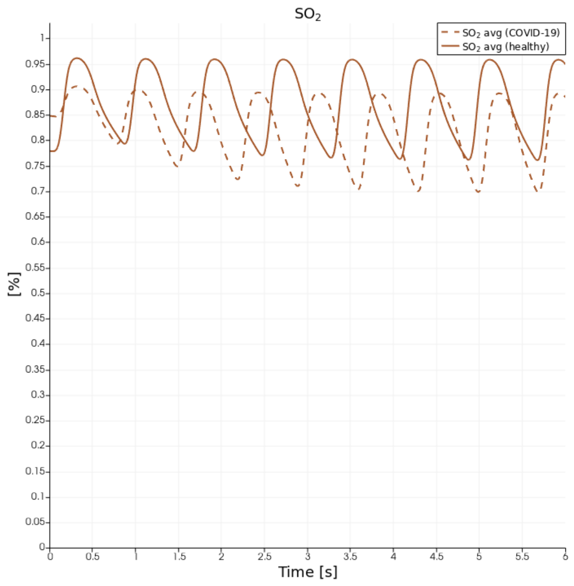
Variable	Measure Unit	Physiological value	Pathological value
$SO_2{}^a$	%	0.98	0.92
$[Hb^*]$	mol m^{-3}	10	8.26
$\tilde{\xi}_0$	mmHg s^{-1}	40	40
T_{HB}	s	0.8	0.7

Table 5.1: List of parameters.

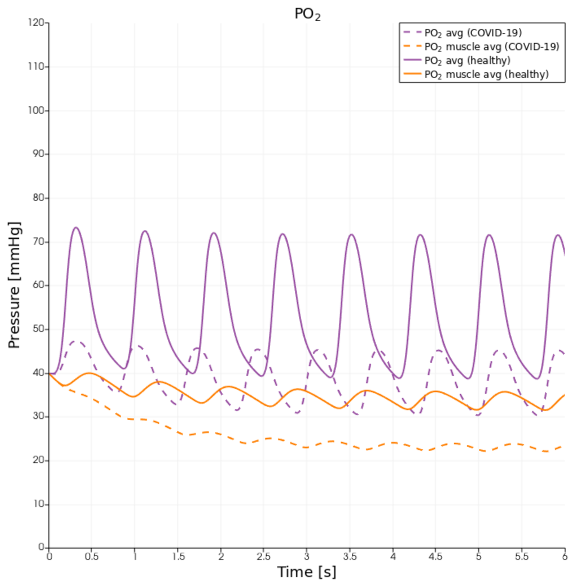
ventilation to the patient, while the target saturation level for patient affected by COVID-19, provided by the NIH (National Institutes of Health), is of $92\% - 96\%$ [5].
The value of $[\mathrm{Hb}^*]$ has been selected as the mean value of Hb concentration in COVID-19 patients computed in the analysis of the Spedali Civili di Brescia, same as the value of the heartbeat period, taken from the same work in which the researchers observed a mean value of cardiac frequency of 85 bmp. Notice that both the values of hemoglobin concentration of healthy and COVID-19 patient are lower than the normal threshold, this is due to the fact that the values are taken from real case in which the patients were probably old or affected by comorbidity.
Finally, the value of the maximum consumption rate in Michaelis-Menten dynamics is assumed to be equal in pathological and physiological condition.
We can see the results of the comparison in Figure 5.2.
Regarding the SO_2, we can see how in physiological condition the saturation oscillates between $70\% - 95\%$, while in pathological conditions between $70\% - 89\%$, hence in a COVID-19 patient the curve is lower than for a healthy one.
The plots regarding PO_2 show a lower magnitude of the curves in pathological case for both the quantities considered: this confirms the reduction of partial pressure observed in COVID-19 patients. On the other hand, the oxygen partial pressure in the muscle is not easily measurable on real case, hence, although its reduction makes physically sense in a pathological patient, we have not a corresponding result in literature.
The concentration of oxygen in the pathological case shows a lower curve than in physiological case, but with a similar trend. This factor is indeed influenced by the differences in the saturation and hemoglobin concentration, parameters which have lower values for a COVID-19 patient.
Finally, regarding the fluxes of oxygen transported from the capillaries to the tissue and used from muscle to produce ATP, we can notice a reduction in both the quantities for a patient affected by COVID-19, difference which becomes larger and larger the more simulation time we considered.
Moreover, notice that of course all this quantities oscillates with a higher frequency, since we simulated considering a higher heartbeat frequency.

5.3 Effect of physical activity and COVID-19 disease

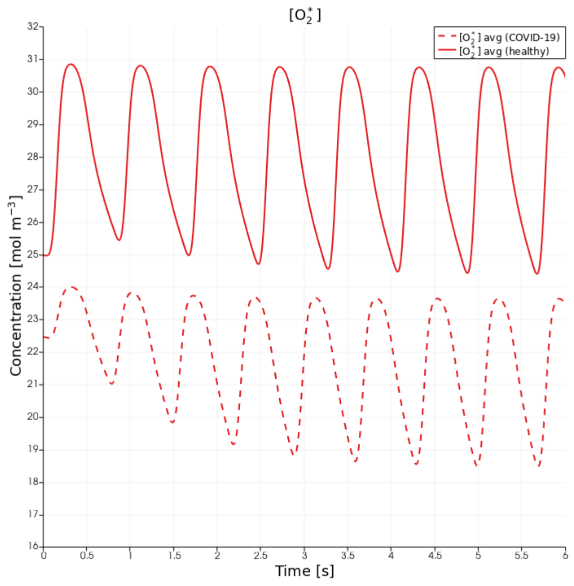
Finally, we compare both the previous cases with the case of a patient affected by COVID-19 disease and not at rest, as before, but under physical activity.
We can see the results of these comparison in Figure 5.3.



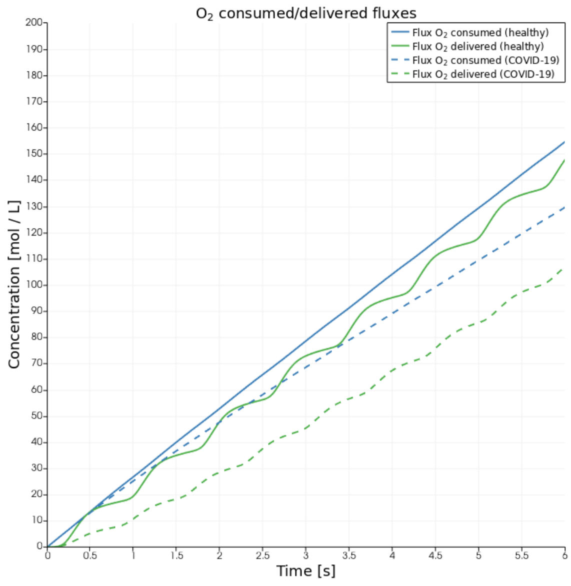
(a) SO$_2$.



(b) PO$_2$ and PO$_2{}^m$.

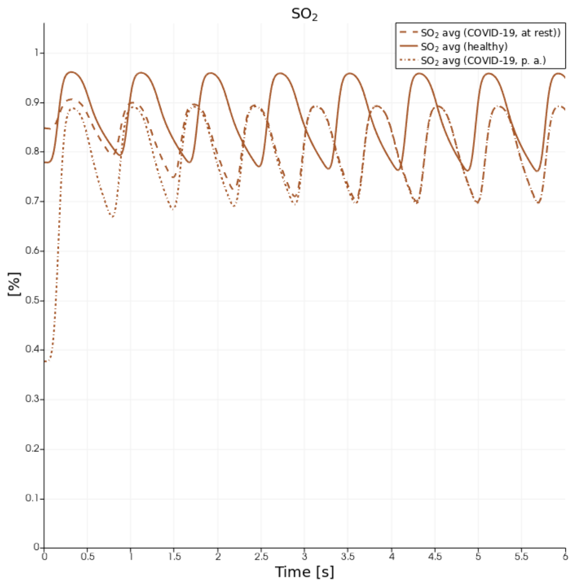


(c) [O$_2^*$].

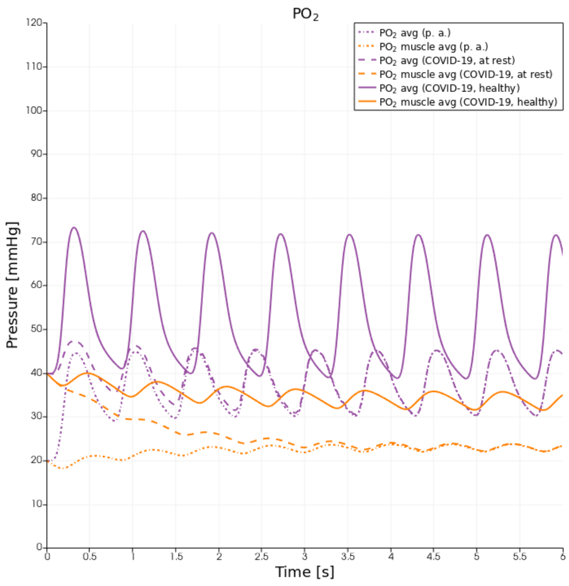


(d) O$_2$ consumed and delivered fluxes.

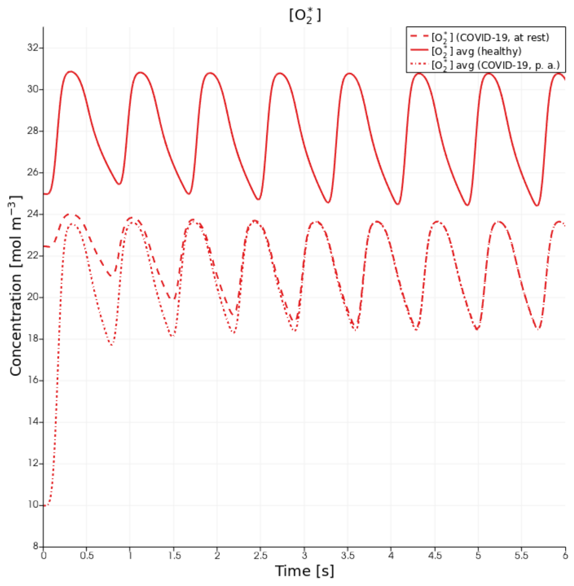
Figure 5.2: Comparison between healthy patient and patient affected by COVID-19 disease, $l = 10^{-1}$ m and with $\Delta t = 10^{-2}$ s.



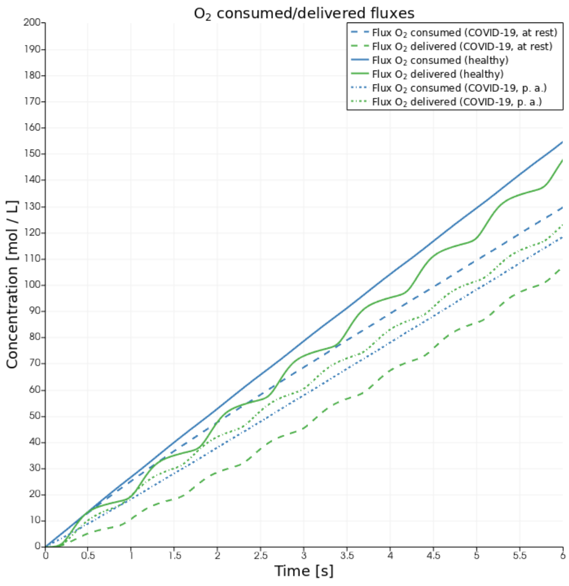
(a) SO_2.



(b) PO_2 and $PO_2{}^m$.



(c) $[O_2^*]$.



(d) O_2 consumed and delivered fluxes.

Figure 5.3: Comparison between healthy patient, patient affected by COVID-19 disease at rest and affected by COVID-19 disease under physical activity, $l = 10^{-1}$ m and with $\Delta t = 10^{-2}$ s.

In particular, with respect with a COVID-19 patient, we can see that, as expected, has a considerable effect on SO_2, PO_2, ${PO_2}^m$ and $[O_2^*]$, but only in the initial phase of the simulation. Indeed, the curves start at different values, but converges to ones of the COVID-19 patient. Also in this case we have more delivered oxygen, as explained before. This situation can be particularly dangerous for a COVID-19 patient, because a patient apparently in good condition can suddenly go under an oxygen deficit if he starts performing physical activity.

Chapter 6

Conclusions and further development

In this work we showed a new OE model, providing its mathematical formulation as it is and under a quasi static approximation, to make the model more computationally affordable. The model was discretized in 0D and 3D cases and then implemented inside life^x library,; one can test them through an app, in which the problem is solved by a Runge-Kutta scheme and the FEM respectively. Moreover, our model was coupled one-way to other different physics to make it more useful in the life^x context and according to iHeart aim. This half-coupling model can be tested through an example implemented inside the library.
Thanks to the tests in Chapter 4, we could validate our implementation of the model: in particular, we checked the coupling with the other physics since it involved the using of different classes already provided in the library, but not fully suitable or immediately usable for the test we could make.
Finally, we used the model to make simulation of real cases, discussing the goodness of the model and studying the effect of some particular conditions, i.e. being under physical activity or affected by COVID-19 disease, on the patient.
This model was developed very recently and it can be improved, as already planned to do in the future. For example, instead of Michaelis-Menten muscle dynamics, the OE model could be coupled with another one for the cellular metabolism.
Moreover, regarding the coupling with perfusion model, it will be interesting to use a real inflow source given by the fluid dynamics instead of the one, spatially homogeneous, we provided. In this way, we can see how the coronary provides blood flux into a volume representing a real heart and how the oxygen concentrations and pressure computed by OE change accordingly in space. As we have modified Perfusion class and implemented our example, it could be easily turned on the coronaries model simulation inside it. For our validation was convenient to keep them off.
Finally, concerning the coupling with other physics, we could also couple two-way instead of one-

way, meaning that the results of OE model could be pass as input to other models to improve their physical accuracy. As an example, a fluid dynamics problem, instead of considering as optimal the value of oxygen concentration in the blood, it could use the estimated one provided by OE, performing in this way a more accurate simulation with fluid viscosity better estimated. Our main effort was to add a completely new class and objects to test its implementation (apps and example) as more similarly to and compatible with the other physics implemented inside the library as possible. Participating in **life**^{x} development indeed required adhering to the coding guidelines of the developers community. Therefore, as part of the project work, we learned to use effectively git version control, automated testing pipelines and automatic indentation software.

Bibliography

[1] F. Regazzoni – *Oxygen Exchange model*, internal report, Politecnico di Milano (2020).

[2] P. C. Africa, R. Piersanti, M. Fedele, L. Dede', A. Quarteroni – *life^x - heart module: a high-performance simulator for the cardiac function*, https://arxiv.org/abs/2201.03303, Politecnico di Milano (2022).

[3] S. Di Gregorio, M. Fedele, G. Pontone, A. F. Corno, P. Zunino, C. Vergara, and A. Quarteroni – *A computational model applied to myocardial perfusion in the human heart: From large coronaries to microvasculature*, Journal of Computational Physics 424 (2021) 109836.

[4] R. M. Inciardi, M. Adamo, L. Lupi, D. S. Cani, M. Di Pasquale, D. Tomasoni, L. Italia, G. Zaccone, C. Tedino, D. Fabbricatore et al. – *Characteristics and outcomes of patients hospitalized for covid-19 and cardiac disease in northern italy*, European heart journal, vol. 41, no. 19, pp. 1821–1829, 2020.

[5] N. Shenoy, R. Luchtel, and P. Gulani – *Considerations for target oxygen saturation in covid-19 patients: are we under-shooting?*, BMC medicine, vol. 18, no. 1, pp. 1–6, 2020.

[6] A. Quarteroni, R. Sacco, F. Saleri, P. Gervasio – *Matematica Numerica*, Springer, 4a edizione.

[7] life^x documentation – https://lifex.gitlab.io/lifex/index.html

[8] deal.ii deocumentation – https://dealii.org/

[9] iHeart project website – https://iheart.polimi.it/en/home/